Réalité augmentée et intelligence artificielle pour une chirurgie hépatique guidée par échographie intravasculaire

Sidaty El Hadramy

Université de Strasbourg

École doctorale Mathématiques, sciences de l'information et de l'ingénieur | ED 269 Université de Strasbourg

UNIVERSITÉ DE STRASBOURG

École Doctorale de Mathématiques,

Sciences de l'Information et de l'Ingénieur

Laboratoire des sciences de l'ingénieur, de l'informatique et de l'imagerie - UMR 7357

THESIS

presented for the grade of

Docteur de l'Université de Strasbourg

by

Sidaty El hadramy

AI-ENABLED IVUS-GUIDED AUGMENTED REALITY FOR LIVER SURGERY

Publicly defended on **December 2, 2024**

Members of the jury:

Mr. Hervé Delingette ..... **Reviewer**
Research Director, Inria, France.

Mr. Yohan Payan ..... **Reviewer**
Research Director, CNRS, France.

Mrs. Caroline Essert ..... **Examiner**
Professor at the University of Strasbourg, France.

Mr. Ole Jakob Elle ..... **Examiner**
Research Director at the Oslo University Hospital, Norway.

Mr. Stéphane Cotin ..... **Supervisor**
Research Director, Inria, France.

Mr. Nicolas Padoy ..... **Supervisor**
Professor at the University of Strasbourg, France.

Université de Strasbourg

Sidaty El hadramy

École doctorale Mathématiques, sciences de l'information et de l'ingénieur | ED 269
Université de Strasbourg

Résumé

Cette thèse présente un nouveau système de réalité augmentée (RA) guidé par l'échographie intravasculaire (IVUS) destiné à améliorer la précision des interventions chirurgicales hépatiques mini-invasives. Bien que l'IVUS offre des avantages pour l'imagerie intraopératoire, aucune méthode antérieure n'a intégré ces données en temps réel dans le guidage chirurgical des procédures hépatiques. La méthode proposée vise à aligner les données IVUS intraopératoires avec les images CT préopératoires, offrant ainsi aux chirurgiens une vue anatomiquement précise des structures hépatiques, adaptée à la déformation de l'organe durant l'intervention. Pour la mise en place de ce system, nous adoptons une approche de recalage non-rigid basée sur les caractéristiques anatomiques, combinant un modèle biomécanique avec des modèles d'intelligence artificielle (IA). Cette combinaison permet d'assurer à la fois la précision et le fonctionnement en temps réel. Les contributions majeures comprennent le développement du SOniCS, intègrant FEniCS et SOFA pour une modélisation précise des tissus mous, ainsi qu'une méthode alimentée par l'IA permettant de reconstruire des volumes IVUS 3D sans systèmes de suivi externes, simplifiant ainsi la configuration chirurgicale. La contribution principale est la méthode de recalage spécifique au patient et fonctionnant en temps réel, permettant un alignement continu des données IVUS avec les modèles préopératoires et une adaptation aux caractéristiques individuelles des patients. Le travail présenté dans cette thèse constitue une avancée significative dans le domaine de la chirurgie hépatique assistée par ordinateur, offrant un système pratique pour assister les interventions chirurgicales hépatiques mini-invasives.
Mots clés : Intelligence Artificielle, Réalité Augmentée, Chirurgie Hépatique

Abstract

This thesis presents a novel augmented reality (AR) system guided by intravascular ultrasound (IVUS) to enhance the precision of minimally invasive liver surgeries. While IVUS provides advantages for intraoperative imaging, no previous method has integrated this real-time data into the surgical guidance of liver procedures. The proposed method aims to align intraoperative IVUS data with preoperative CT images, providing surgeons with an anatomically accurate view of liver structures that adapt to the deformation of the organ during the intervention. We adopt a non-rigid registration approach based on anatomical features to implement this system, which combines a biomechanical model with artificial intelligence (AI) models. This combination allows for both accuracy and real-time. Major contributions include the development of SOniCS, a framework that integrates FEniCS and SOFA for accurate soft tissue simulation, and an AI-driven method for reconstructing 3D IVUS volumes without external tracking systems, thus simplifying the surgical setup. The primary contribution of this thesis is developing a patient-specific, real-time registration method that allows for continuous alignment of IVUS data with preoperative models and adaptation to patient-specific characteristics. The work presented in this thesis represents an advancement in computer-assisted liver surgery, providing a practical system to assist minimally invasive liver interventions.
Key words : Artificial Intelligence, Augmented Reality, Liver Surgery

جدتي الغالية...

لقد كنتِ دائمًا مثالًا حيًا للإرادة القوية والعزيمة التي لا تلين، وكان عملكِ المستمر دون كلل أو ملل من أجل توفير حياة كريمة مصدر إلهام كبير لي. كنتِ تواجهين صعوبات الحياة بشرف، وتبذلين جهدكِ بكل فخر من أجل تأمين مستقبل أفضل لنا.

من خلالكِ، تعلمت معنى المثابرة والإصرار، وكيف أن العمل الجاد هو السبيل الوحيد لتحقيق أي هدف. لقد كان كفاحكِ مصدر قوتي. تعلمتُ منكِ أن الحياة لا تتوقف عند العثرات، وأنه بالعزيمة والإصرار يمكن التغلب على أي تحدي.

رغم غيابك عن عالمنا، فإن ذكراكِ ستظل عالقة في قلبي وعقلي، وأنتِ لا تزالين حية في كل خطوة أخطوها في هذا الطريق. كل إنجاز أحققه، وكل عمل أتمه، هو امتداد لما غرستِه فيَّ من قيم، صبر وإصرار.

اللهم ارحم التبتية منت سيد أحمد ولد أحمد دينة، وأسكنها فسيح جناتك.

ACKNOWLEDGMENTS

Je tiens à exprimer ma plus profonde gratitude à mon directeur de thèse, Stéphane Cotin, dont le soutien sans faille a été indispensable tout au long de cette aventure. Il ne m'a pas seulement offert l'opportunité de poursuivre et de mener à bien ce projet de thèse, mais il m'a également permis de vivre une expérience de vie véritablement transformative. Stéphane, ton mentorat a été une source d'inspiration inestimable, et je suis profondément reconnaissant de la confiance que tu as placée en moi dès le début. Merci pour tes encouragements constants, tes conseils éclairés et ta patience exceptionnelle. Ta capacité à m'écouter attentivement et à me guider à travers les défis de ce projet a été un atout précieux.

Je tiens également à exprimer ma sincère gratitude à Nicolas, mon second superviseur, pour son expertise, son soutien précieux et sa disponibilité tout au long de ce parcours. Son approche rigoureuse et ses conseils perspicaces ont grandement contribué à l'avancement de ce travail.

I would like to sincerely thank the members of my thesis committee: Hervé Delingette, Yohan Payan, Caroline Essert, and Ole Jakob Elle. I am particularly grateful to the two reviewers, Hervé Delingette and Yohan Payan, for their time and insightful feedback. It is truly an honor to have my thesis read and evaluated by such esteemed experts, especially considering their demanding schedules. Their contributions have been invaluable in enhancing the quality of this work.

I would like to sincerely thank Juan Verde, who has been a valuable clinical supervisor throughout this thesis. His advice, ideas, and passion for artificial intelligence have significantly contributed to the development of this work. Thank you for all your encouragement and for always believing in me. Your support has been essential to my progress, and I am deeply grateful for everything you've done.

Je tiens à adresser mes sincères remerciements à tous les membres de l'équipe MIMESIS Inria pour leur aide précieuse tout au long de ce travail. Les discussions scientifiques enrichissantes et stimulantes ont grandement contribué à l'avancement de ce projet et à mon développement personnel. Au-delà des échanges intellectuels, ce sont également les moments de convivialité qui resteront gravés dans ma mémoire. Les pauses animées par des parties de Coiche, Tarot, SkyJo et bien d'autres jeux ont apporté une belle dose de détente et de bonne humeur tout au long de cette aventure.

Je souhaite citer en particulier Robin Enjalbert, Valentina Scarponi, Vincent Italiano, Nicola Zotto, François Lecomte, Karl-Philippe Beaudet, Pierre Galmiche, Pablo Alvarez, Thomas Wahl, et Guillaume Mestdagh avec qui j'ai eu la chance de partager de nombreux

moments, tant professionnels que personnels. Leur soutien, leur camaraderie et leur enthousiasme ont été d'une grande aide pendant cette période. Merci !

Je tiens à rendre un hommage particulier à mon père, Brahim El Hadramy, qui a toujours eu une foi inébranlable en moi et m'a constamment poussé à aller de l'avant. Sans lui, je n'aurais jamais osé rêver de poursuivre une thèse ni d'en arriver là où je suis aujourd'hui. Son soutien sans faille, tout au long de ma vie, a été mon plus grand atout. Il a toujours été mon ange gardien, veillant sur moi avec une sagesse et une bienveillance constantes. À chaque étape de ce parcours, il a été l'épaule solide sur laquelle je pouvais me reposer, peu importe les obstacles. Grâce à lui, j'ai appris la persévérance, la confiance en moi et l'importance de croire en mes rêves, même lorsque les défis semblaient insurmontables. Merci, papa, pour ton amour inconditionnel et pour avoir été mon guide tout au long de cette aventure.

Je tiens à remercier mon frère, Mohamed Yahya El Hadramy, pour sa force, son soutien indéfectible et sa présence constante tout au long de ce parcours. Ton encouragement et ta bienveillance ont été une source d'inspiration constante, m'aidant à surmonter les moments les plus difficiles. J'espère que ce travail pourra à son tour t'inspirer à terminer bientôt ton doctorat, car je ne doute pas de ta capacité à accomplir de grandes choses.

Je souhaite exprimer toute ma gratitude et mon amour à ma mère, Hindou Lekhdeyem, ainsi que mes freres et soeurs, qui a toujours cru en moi, même dans les moments les plus difficiles. Votre soutien constant, vos paroles pleines de réconfort et votre capacité à toujours trouver les mots justes, malgré la distance qui nous sépare, ont été pour moi une source inépuisable de force et de motivation.

Je tiens à remercier profondément toutes celles et tous ceux qui ont contribué, de près ou de loin, à la réussite de cette thèse. Que ce soit par leurs conseils, leur soutien ou leur présence, chacun a joué un rôle essentiel dans l'accomplissement de ce travail. À toutes celles et tous ceux qui liront ces lignes sans voir leur nom mentionné, je demande humblement de bien vouloir me pardonner, car leur contribution est tout aussi précieuse et ne saurait être oubliée.

ACRONYMES

ADR	Average Drift Rate
AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
B-mode	Brightness Mode
BCs	Boundary conditions
CAD	Coronary Artery Disease
CCT	Contrasted Computed Tomography
CNN	Convolutional Neural Network
CPD	Coherent Point Drift
CRC	Colorectal Cancer
CT	Computed Tomography
ConvLSTM	Convolutional Long Short Term Memory
DL	Deep Learning
DOFs	Degrees of Freedom
ECC	Entropy Correlation Coefficient
FDR	Final Drift Rate
FE	Finite Element
FEM	Finite Element Method
FFC	FEniCS Form Compiler
FPS	Frame Per Second
GMM	Gaussian Mixture Model
HCC	Hepatocellular Carcinoma
HV	Hepatic Vein
IBM	Immersed Boundary Method
ICP	Iterative Closest Point
IGS	Image Guided Surgery

IMU	Inertial Measurement Unit
IOUS	Intraoperative Ultrasound
IVC	Inferior Vena Cava
IVUS	IntraVascular Ultrasound
LCC	Linear Correlation of Linear Combination
LLR	Laparoscopic Liver Resection
LUS	Laparoscopic Ultrasound
LSTM	Long Short Term Memory
MAE	Mean Absolute Error
MCC	Multiple Correlation Coefficient
MI	Mutual Information
MIS	Minimally Invasive Surgeries
ML	Machine Learning
MPCECT	Multi Phase Contrast Enhanced Computed Tomography
MRI	Magnetic Resonance Imaging
MSE	Mean Square Error
NCC	Normalized Cross Correlation
NCCT	Non Contrasted Computed Tomography
NICP	Non-Rigid Iterative Closest Point
NMI	Normalized Mutual Information
NR	Newton Raphson
PBDT	Physics Based Digital Twin
PCI	Percutaneous Coronary Interventions
PDE	Partial Differential Equations
PDMS	PolyDiMethylSiloxane
RFA	Radio Frequency Ablation
ReLU	Rectified Linear Unit
SOFA	Simulation Open Framework Architecture
SSD	Sum of Squared Differences
SVK	Saint-Venant Kirchhoff
Seq2Vec	Sequence to Vector
TRE	Target Registration Error
UFL	Unified Form Language
US	Ultrasound
VM	Vessel Map

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Clinical Context

The liver is a highly complex and fascinating organ located in the thoracoabdominal cavity, in direct contact with the diaphragm, which separate both cavities. It is the largest organ in the human body and consists of two lobes divided by a falciform ligament (See Figure 1.1). The liver holds approximately 13% of the body's blood supply at any given time [Meskell (2010)]. This blood primarily enters the liver through the portal vein, carrying nutrients and substances from the gastrointestinal tract, spleen, pancreas, and gallbladder. In addition, the liver receives oxygen-rich blood from the hepatic artery, which branches directly from the celiac trunk and descending aorta.

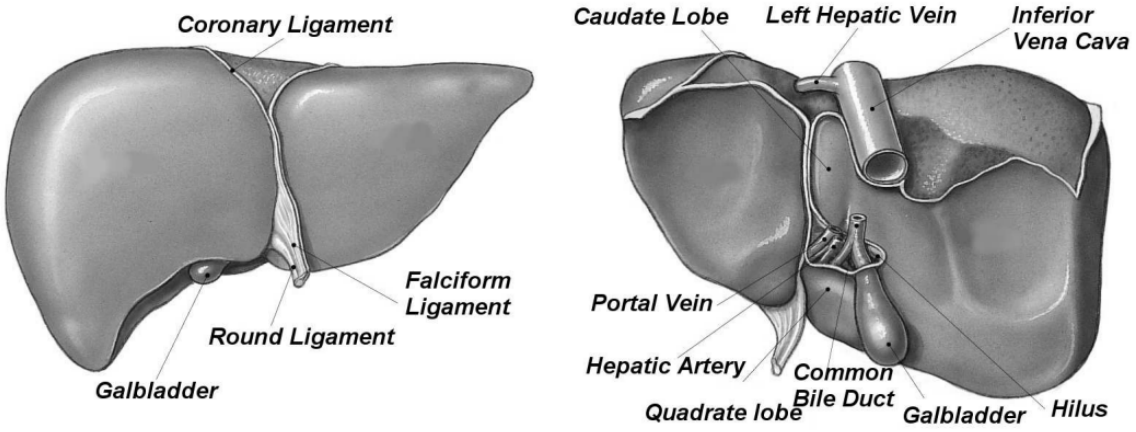


Figure 1.1: Anatomy of the liver. The images are taken from: https://web.unicz.it

Inside the liver, the portal vein, hepatic artery, lymphatics, nerves, and hepatic bile duct converge at a central location within the liver, forming a network that branches out through a structure known as the portal canal, see Figure 1.2. From the portal canal, these pathways extend throughout the liver, culminating in the sinusoids, known as the liver's capillary system. Here, blood from the portal vein combines with blood from the hepatic artery's end-arterial branches. After traversing the sinusoids, the mixed blood flows into the central veins, eventually exiting the liver via the hepatic vein. This process supports vital functions such as filtering toxins from the bloodstream, regulating blood sugar levels, and producing bile for fat digestion. This makes the liver vulnerable to various diseases. These diseases contribute to multiple abnormalities and shorten life expectancy [Landreau et al. (2015)].

Liver cancer, with its rising incidence and high mortality rates, presents a significant global health challenge. It is the fifth most common cancer worldwide and the third leading cause of cancer mortality [Ferlay et al. (2010)]. Hepatocellular carcinoma (HCC), the

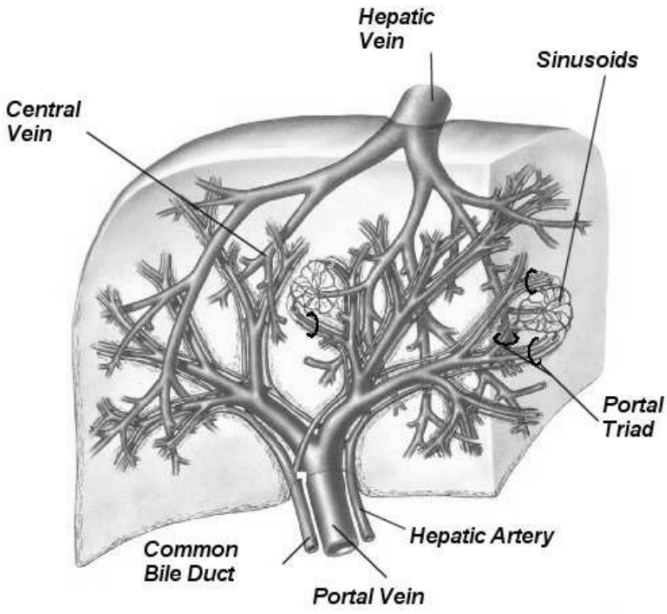


Figure 1.2: Network of interconnecting and diverging blood vessels within the liver. The image is taken from: https://web.unicz.it.

primary form of liver cancer, accounts for 4.7% of all new cancer cases and 8.2% of cancer-related deaths [Bray et al. (2018)]. The liver is also a common site for metastases from other cancers, particularly colorectal cancer (CRC), which is the second most common cancer globally, responsible for 10.2% of new cases [Steele et Ravikumar (1989)]. Each year, around 1,750,000 individuals are expected to be diagnosed with either primary or secondary liver cancer [Bray et al. (2018)], highlighting the urgent need for effective treatment and prevention strategies. Major risk factors include chronic hepatitis B and C infections, cirrhosis, excessive alcohol consumption, and obesity. Liver cancer is characterized by the appearance of abnormal cells that do not perform their usual functions. These abnormal cells form tumors, tissues that vary in size and shape and can develop in any part of an organ.

Depending on the size and location of the tumors, as well as the patient's overall condition, liver cancer can be treated using several methods: liver resection, Radiofrequency Ablation (RFA), or liver transplantation. The appropriate treatment is typically determined through multidisciplinary decision-making, which involves detailed imaging and assessment of the organ's characteristics and the patient's overall health. Liver resection is a surgical procedure that involves removing a segment of the liver containing tumors [Orcutt et Anaya (2018), Galle et al. (2018)]. The primary challenge with this approach is ensuring that the remaining portion of the liver is sufficiently large and receives adequate blood supply to maintain normal liver function. Surgeons carefully assess the patient's

liver health, along with the tumor's location and its proximity to surrounding structures, to ensure that the liver can regenerate and maintain its vital functions after surgery. In RFA, clinicians use heat to destroy cancer cells by inserting a needle-like probe directly into the tumor [Curley et al. (1999)], hence destroying unhealthy cells. This minimally invasive technique is often used for smaller tumors or patients who are not candidates for surgery. Liver transplantation is considered for a more extensive disease [Llovet (2005)]. This procedure involves replacing the diseased liver with a healthy liver from a donor, offering a potential cure for patients with advanced liver cancer. For HCC, liver transplantation is often the preferred treatment due to its potential to remove both the tumor and any underlying liver disease simultaneously. However, the scarcity of compatible donors frequently makes transplantation impractical for many patients, leading to liver resection as the more viable alternative. In contrast, for CRC that has metastasized to the liver, resection is generally the preferred approach. This preference is because CRC metastases often involve discrete, localized tumors that can be effectively removed while preserving liver function.

1.2 Liver surgery

Liver surgery, particularly hepatectomy or liver resection, is the single and only treatment with curative intentions. This surgical procedure involves removing the tumor along with a margin of healthy tissue, aiming to eradicate cancer cells and prevent recurrence [Orcutt et Anaya (2018)]. The generative capacity of the liver allows it to recover and regrow after significant portions are excised, making surgery a viable option for many patients. A successful surgical resection necessitates the complete removal of the tumor(s) with safety margins while preserving as much healthy tissue as possible. Additionally, vital liver parts, such as arteries and veins, must be spared or strategically divided during the surgery. These requirements make resection procedures highly complex from both technical and clinical perspectives. Before the intervention, an initial contrasted 3D medical imaging method, such as CT or MRI, is usually acquired from the patient's abdomen area. This 3D image, called preoperative, helps clinicians localize the tumors' positions, evaluate the difficulty of the intervention, and plan the resection if needed. Depending on the patient's condition, the surgery can be categorized into two main types: **open** and **minimally invasive** (including laparoscopic and robotic), see Fig 1.3. The choice between these approaches depends on several factors, including the tumor's size and location, the patient's overall health, and the surgeon's expertise [Mirnezami et al. (2011)]. These considerations help determine the most appropriate surgical method to achieve the best pos-

sible outcome.

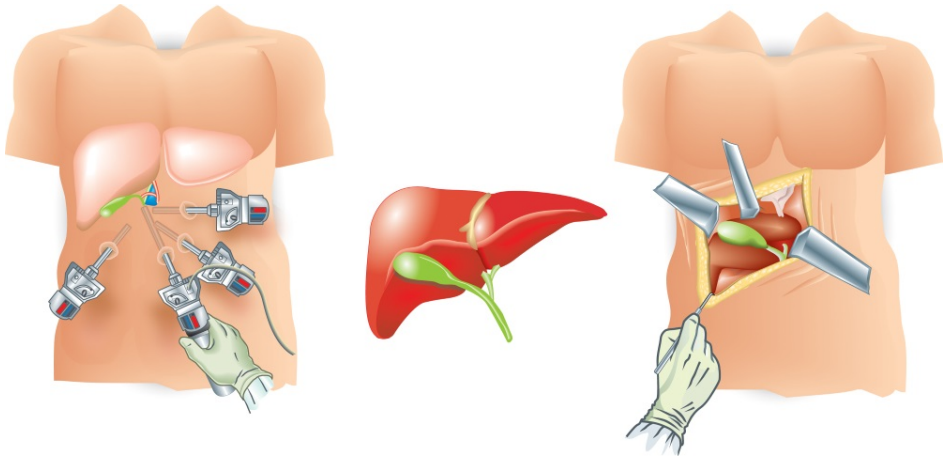


Figure 1.3: Surgical approaches to liver resection: On the left, the laparoscopic approach is depicted, involving small incisions through which laparoscopic tools are inserted to perform the surgery. The open surgery approach is illustrated on the right, where a large incision is made to access the liver directly. The images are taken from: https://arinmed.com/

Open liver surgery is the traditional approach, where a large incision is made in the abdomen to access the liver directly. This method provides the surgeon with a clear and comprehensive view of the liver and surrounding structures, allowing for extensive procedures and thorough tumor removal. The advantages of open surgery include its ability to address more significant and complex tumors and the surgeon's greater access to the liver for precise manipulation [Wang et al. (2019)]. However, open surgery also has notable drawbacks, such as a longer recovery time, increased risk of postoperative pain and infection, and a more visible scar. These factors can impact the patient's overall recovery experience and time away from daily activities.

Laparoscopic liver resection (LLR), or minimally invasive liver surgery, is characterized by the use of small incisions through which specialized instruments and a camera are inserted to operate the patient [Reich et al. (1991)]. LLR offers significant advantages over traditional open liver surgery. The minimally invasive nature of the procedure, which involves smaller incisions, leads to reduced postoperative pain and faster recovery times for patients. This results in shorter hospital stays and a quicker return to normal activities and work. Additionally, LLR typically causes less blood loss during surgery, reducing the need for blood transfusions and minimizing the risk of related complications [Wang et al. (2019)]. However, LLR has notable drawbacks that can impact its effectiveness and suitability for some patients, as reported by several studies [Cherqui et al. (2000), Dulucq et al. (2005)]. One major drawback of LLR is the limited access to the liver and surrounding structures. The small incisions restrict the surgeon's ability to maneuver and visualize

the entire liver comprehensively. This limitation can make it challenging to address large or complex tumors effectively, as the surgeon has less room to work and may not be able to see or reach certain areas of the liver fully. Another significant drawback is the technical complexity involved in LLR. The use of specialized instruments and the laparoscopic camera requires a high degree of precision and skill. Surgeons must navigate the limited space and operate with a 2D view of the liver, which can be less intuitive than the direct 3D view obtained during open surgery. A further drawback is the lack of haptic feedback during laparoscopic procedures. Unlike open surgery, where surgeons can directly feel the tissues and organs they are working on, laparoscopic surgery relies on visual cues and instrument resistance. This absence of tactile sensation can make it more challenging to gauge the force needed, increasing the risk of inadvertent damage to healthy tissue or vital structures. The lack of haptic feedback necessitates greater reliance on visual information, which can sometimes be insufficient for ensuring the delicate handling required in complex liver surgeries.

Due to these drawbacks, accessing tumors outside the anterior periphery of the right lobe or the left lobe is challenging [Descottes et al. (2003)]. Tumors in these regions are typically considered high-risk for resection because laparoscopic tools may not easily reach them, and major vessels nearby could cause life-threatening blood loss if damaged. Image-Guided Surgery (IGS) is a modern surgical technique that leverages advanced imaging technologies to assist surgeons in performing precise and minimally invasive procedures [Blau et al. (2016)]. In the context of LLR, IGS is invaluable due to the liver's complex vascular structure and its proximity to other vital organs. It allows surgeons to visualize and navigate the anatomy in real-time, improving accuracy and reducing the risk of complications. One emerging form of IGS is augmented reality (AR). AR superimposes the 3D preoperative models of the patient's liver, including tumors and blood vessels, onto the surgeon's view, providing a detailed roadmap during the operation [Acidi et al. (2023)]. This integration improves spatial orientation and situational awareness, thus improving real-time decision-making. Making locating and removing liver lesions easier while preserving healthy tissue precisely.

1.3 Augmented reality

Augmented reality (AR) is a method that can potentially enhance the precision of LLR [Brunet et al. (2019); Haouchine et al. (2013)]. Preoperative images, such as CT or MRI scans, are used to create a detailed three-dimensional model of the liver, including its internal structures. During the preoperative planning stage, surgeons segment and identify

the organ and its internal structures, including tumors. This three-dimensional model, complete with marked tumors, can then be integrated with the intraoperative view [Feuerstein et al. (2007)] to provide AR. By incorporating AR, clinicians can visualize the liver model overlaid with the real-time surgical field during surgery. This integration allows them to accurately identify the current positions of the tumors on the organ, facilitating precise resection. The use of AR in LLR would be beneficial as it compensates for the limited field of view inherent in minimally invasive procedures. AR provides surgeons with a comprehensive, enhanced visualization that improves their ability to navigate and operate within the complex anatomy of the liver. An example of AR for LLR, from Haouchine et al. (2013), is illustrated in Figure 1.4, where the 3D model of the liver, portal and hepatic veins, and the tumors are overlaid onto the laparoscopic view.

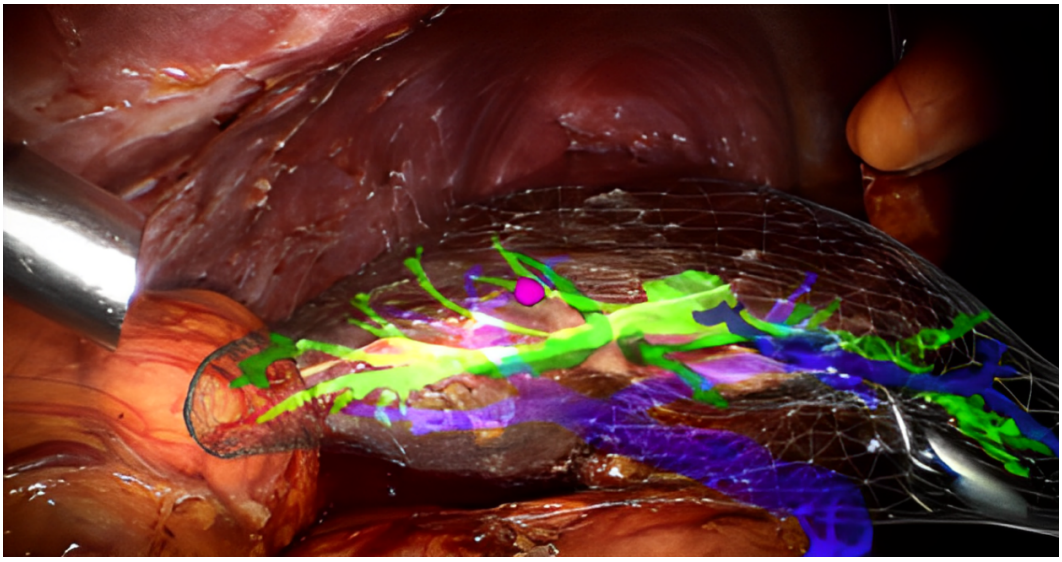


Figure 1.4: An example of Augmented reality (AR) in laparoscopic liver resection (LLR) involves using a preoperative CT scan to create a 3D model of the liver and its internal structures. This model is then overlaid on the intraoperative view to assist surgeons during the procedure. The image is from Haouchine et al. (2013).

The soft tissue nature of the liver and the patient's respiration, combined with the surgeon's interactions with the organ, cause significant deformation during surgery. This dynamic environment means that the preoperative CT images often no longer match the current shape and position of the liver and its structures. The liver moves and deforms as the patient breathes, and surgical manipulations further alter its shape. These changes make it challenging to maintain the accuracy of the preoperative model during the actual procedure. Consequently, the preoperative images can become misaligned with the liver's real-time anatomy, reducing their effectiveness in accurately tracking tumors and internal structures. This misalignment presents a significant challenge, as surgeons cannot

rely on the preoperative model for precise guidance. Discrepancies between the preoperative images and the liver's current state can lead to difficulties in accurately locating tumors and navigating around critical structures. Without real-time updates, the risk of incomplete tumor removal or accidental damage to healthy tissue increases. To address this issue, the preoperative model can be enhanced with intraoperative imaging techniques that provide real-time updates on the liver's deformation and internal structure movements. Integrating modalities such as intraoperative CT, MRI, fluoroscopy, laparoscopic cameras, and ultrasound can dynamically adjust the preoperative model to reflect the liver's current position and shape, offering more accurate and up-to-date information [Oliveira et Tavares (2014)]. This approach enables the AR to be updated during the intervention regarding organ deformation, surgeon interactions, and human respiration.

Enhancement of the preoperative model through integration with intraoperative imaging techniques is achieved via a process known as **registration**. Registration involves aligning and fusing the preoperative data with intraoperative data to create a coherent and updated organ representation. This process ensures a dynamic adjustment of the preoperative model to accurately reflect the liver's current anatomy by correcting for any deformation or movement that has occurred. The precision of the registration is often highly correlated with the choice of intraoperative imaging modalities [Zhang (2021)]. High-quality intraoperative data permits an accurate alignment of preoperative models with the liver's current anatomy, ensuring precise tumor localization and navigation. Intraoperative CT and MRI are highly effective, offering detailed and high-resolution images. However, they have drawbacks such as radiation exposure, increased procedural time, and higher costs and logistical complexities [Shekhar et al. (2010)]. Fluoroscopy provides real-time imaging but is limited by radiation exposure and offers less detailed soft-tissue contrast. Laparoscopic cameras are helpful for direct visualization but are constrained by a restricted field of view and challenging depth perception, providing no information on internal structures [Madhok et al. (2022)]. IntraOperative UltraSound (IOUS), though it has lower image resolution than other cited modalities, is effective for real-time imaging and provides in-depth information about the liver's internal structures. Its major advantages include the absence of radiation exposure, making it safer for patients, and its ability to deliver immediate feedback during surgery. These features make IOUS a practical and valuable tool in clinical settings, particularly for tracking the liver's internal structures. It can offer real-time data to support an accurate registration of preoperative models with the liver's current anatomy [Nakamoto et al. (2007); Shahin et al. (2014)].

1.4 Intraoperative Ultrasound (IOUS)

Ultrasound imaging is a medical technique that uses high-frequency sound waves to produce images of structures within the body. The physics involves sending sound waves into the body using a transducer, which also receives the echoes that bounce back from tissues. These sound waves are generated by piezoelectric crystals within the transducer that vibrate when an electric current is applied, creating pressure waves propagating through the body. A process known as beam forming is employed to utilize these waves effectively. Beamforming directs the waves in a specific pattern to focus on a particular area, enhancing image resolution and clarity. The returning echoes are captured by the transducer and converted back into electrical signals. In B-mode (brightness mode) imaging, these signals are processed to create a two-dimensional grayscale image, where each pixel's brightness corresponds to the echo's amplitude, representing different tissue densities and structures within the scanned area. This method provides detailed cross-sectional images, aiding in diagnosing and monitoring various medical conditions. Placing an ultrasound probe in contact with the tissue of interest allows the measurement of reflected sound waves to produce an image of the underlying anatomy. Different tissue properties, particularly at organ boundaries, determine the contrast in the image.

The use of ultrasound imaging in intraoperative interventions was introduced in the late 1980s [Castaing et al. (1985); Machi et al. (1987)] to enhance the accuracy and safety of these procedures. IntraOperative UltraSound (IOUS) has been particularly valuable in open liver surgery, allowing surgeons to visualize deeply located tumors within the liver parenchyma and identify critical surrounding blood vessels. In addition to open surgeries, ultrasound technology has been adapted for minimally invasive surgeries (MIS). Using ultrasound in these settings is especially important to compensate for the drawbacks of MIS surgeries presented earlier. To support the requirements of MIS, specialized ultrasound probes have been developed, including laparoscopic ultrasound (LUS) and intravascular ultrasound (IVUS), each designed to meet the needs of different types of procedures.

1.4.1 Laparoscopic Ultrasound (LUS)

Building on the success of IOUS in open surgeries, Laparoscopic Ultrasound (LUS) (an example of a LUS probe is presented in Figure 1.5) was developed to extend these benefits to minimally invasive procedures. Introduced by Fukuda et al. (1984), LUS involves coupling an ultrasound transducer to a standard laparoscopic tool, enabling the surgeon to maneuver the probe to scan the liver tissue while observing the laparoscopic video feed,

see Figure 1.6. The transducer is typically mounted on a flexible tip that can be articulated in multiple directions (see Figure 1.5), facilitating comprehensive liver scanning. LUS has become especially important in laparoscopic staging procedures, where it helps assess tumor resectability without large incisions. Since laparoscopic techniques lack tactile feedback, LUS provides essential localization sensitivity, often detecting tumors that may not be visible on preoperative scans, thus enhancing surgical planning and outcomes.

Figure 1.5: An example of Laparoscopic Ultrasound (LUS) probe, Model: BK medical 4-way Laparoscopic linear array transducer 8666-RF. The image is taken from https://kenmedsurgical.com.

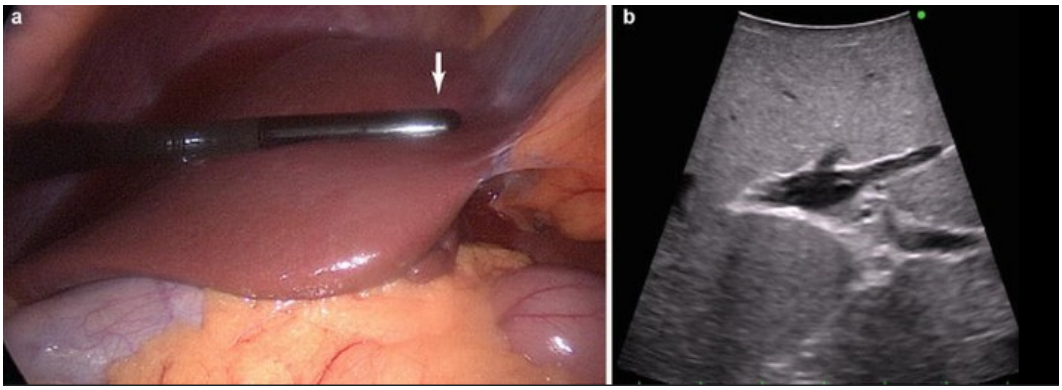


Figure 1.6: Left: Laparoscopic camera image showing the liver, with the LUS transducer in contact with the organ. Right: Image captured using the LUS transducer. Both images are taken from Adams (2014).

However, performing LUS during LLR is particularly demanding due to the combined difficulties of laparoscopy's poor ergonomics, limited field of view, and the skills required to maneuver the probe and analyze images. Additionally, LUS images can be intermittent and asynchronous with the ongoing surgery, necessitating multiple iterations and repetitive steps, such as alternating between inserting the probe and repositioning surgical instruments. Moreover, manipulating the probe can cause liver deformation, further complicating image interpretation and integration in an AR system. Therefore, a method that

allows for continuous and synchronous ultrasound assessment throughout the surgery, with minimal interruptions and without deforming the organ, would offer more potential for registration and AR applications.

1.4.2 IntraVascular Ultrasound (IVUS)

Intravascular ultrasound (IVUS) is an intravascular imaging technique mainly used in interventional cardiology to analyze lesion morphology, measure plaque burden, guide stent sizing, evaluate stent expansion, and detect procedural complications [Mintz et al. (1995)]. It involves the use of a catheter equipped with a miniature ultrasound transducer at its tip (Figure 1.7 illustrates an IVUS transducer). During cardiovascular procedures, the catheter is inserted into the patient's vascular system, typically through the femoral or radial artery, and navigated toward the coronary arteries under fluoroscopic guidance. The catheter is then advanced through the vascular pathway to the target area within the coronary arteries [Parviz et al. (2018)]. Once positioned correctly, the transducer emits sound waves penetrating the vessel walls. These sound waves are reflected to the transducer, which then processes the echoes to create detailed cross-sectional images of the vessel lumen and wall. The IVUS catheter can be manually [Parviz et al. (2018)] pulled back through the artery, allowing for a continuous scan and a view of the vessel's condition. This imaging technique helps cardiologists accurately assess plaque composition, vessel size, and the effectiveness of stent placement, thereby enhancing the precision and safety of cardiac interventions.

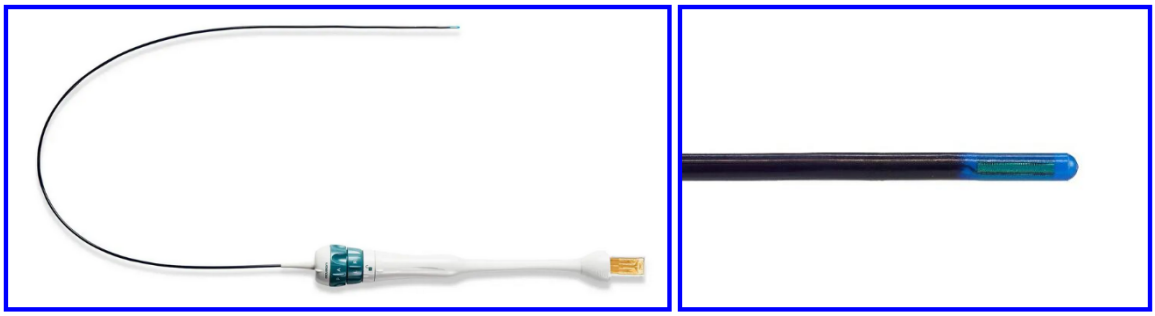


Figure 1.7: Left: An example of IntraVascular Ultrasound (IVUS) probe. Model: ACUSON AcuNav™ Ultrasound Catheter. **Right:** Zoom on the tip where an ultrasound transducer is equipped. Both images are taken from: https://www.jnjmedtech.com/.

IVUS is also finding significant applications in liver surgery, where its minimally invasive and radiation-free imaging capabilities offer substantial benefits for intraoperative guidance. Urade et al. (2021) demonstrated the feasibility of completing the entire navigation of the liver through the Inferior Vena Cava (IVC) and the Hepatic Vein (HV), relying

solely on IVUS images without the need for additional X-ray imaging techniques. As illustrated in Figure 1.8, the catheter is positioned in the lumen of the inferior vena cava. Therefore, an overall and full-thickness view of the liver can quickly and easily be obtained through mostly rotation movements of the catheter, while it is constrained to the lumen of the inferior vena cava and with no interaction with the tissue (contactless).

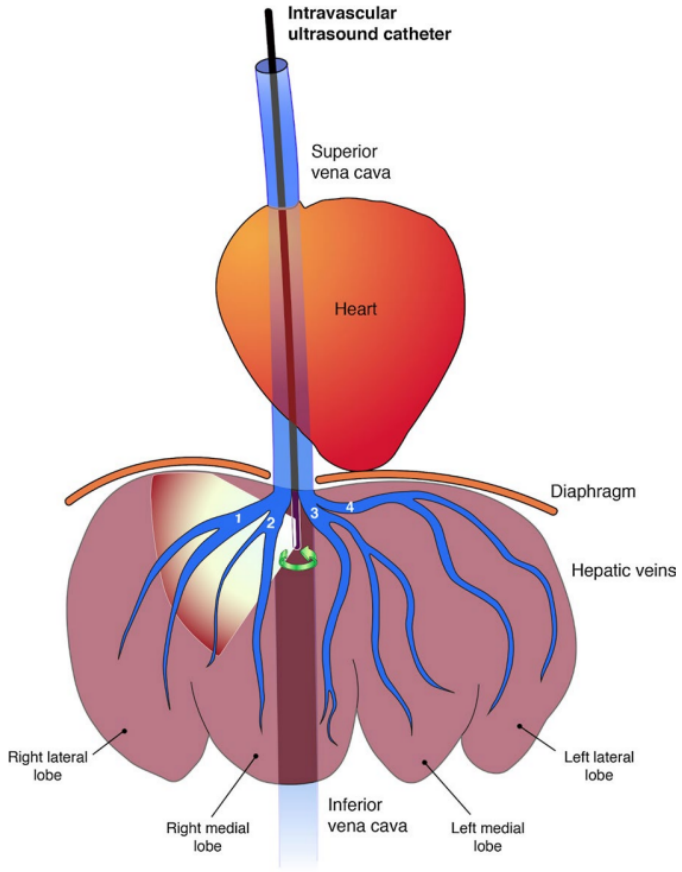


Figure 1.8: Intravascular ultrasound image-guided liver navigation identifying the following structures: **1.** Right hepatic vein; **2.** Right median hepatic vein; **3.** Left median hepatic vein; **4.** Left lateral hepatic vein. The image is taken from Urade et al. (2021).

In contrast to LUS, IVUS offers continuous, synchronous imaging throughout the procedure without occupying one of the incisions used for specialized instruments and the laparoscopic camera. LUS, being in direct contact with the liver's surface, can induce organ deformation and compress adjacent blood vessels, complicating image interpretation and distorting the liver's natural shape. This external pressure can impede the accuracy of the visual data, making it difficult for surgeons to assess the liver's condition and navigate effectively. On the other hand, IVUS is situated within the blood vessels, avoiding direct contact with the liver, and thus does not alter the liver's shape or compress its vessels. It moves in harmony with the liver's natural motion, including respiratory changes,

providing more stable images. This characteristic of IVUS makes it easier to interpret the imaging data, facilitating better surgical decision-making and enhancing the overall effectiveness of liver interventions. The interpretation is demonstrated by Urade et al. (2021) as illustrated in Figure 1.9 where ultrasound images of the hepatic veins are captured by the IVUS positioned at the inferior vena cava.

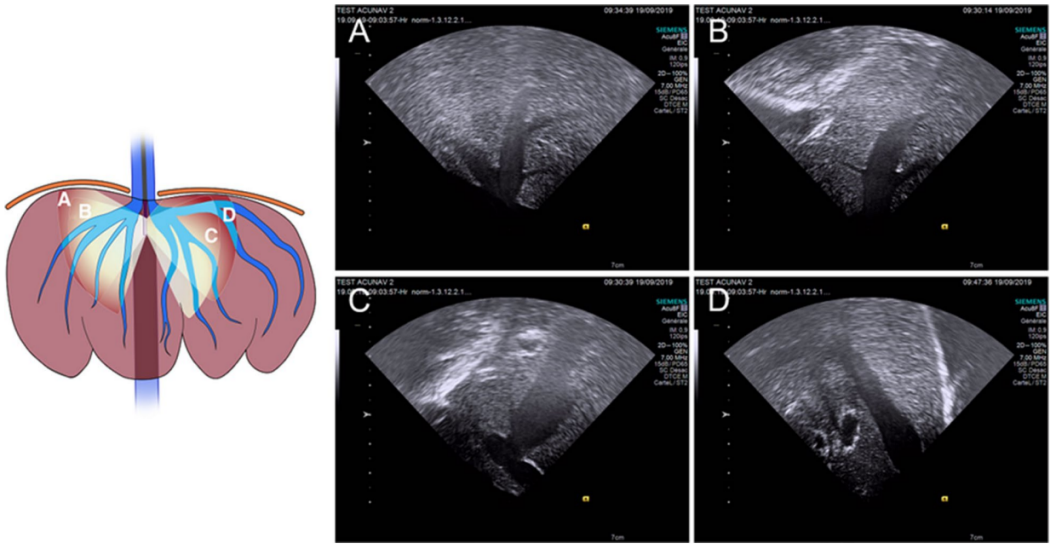


Figure 1.9: Intravascular ultrasound images of the hepatic veins are captured from the inferior vena cava, as depicted in the schematic showing four ultrasound planes labeled A, B, C, and D. In these images, the right side represents the proximal end, and the left side represents the distal end. The specific veins are labeled as follows: **A**: Right hepatic vein, **B**: Right median hepatic vein, **C**: Left median hepatic vein, and **D**: Left lateral hepatic vein. This illustration was created by Urade et al. (2021).

1.5 Ultrasound-guided augmented reality

In the previous sections, we have discussed the advantages that AR may offer in the context of liver surgery, particularly in addressing the limitations of LLR. We presented how augmented reality can enhance the surgeon's ability to visualize and resect tumors accurately, thus overcoming the restricted field of view inherent in laparoscopic surgery. Additionally, we explored the role of IOUS as a valuable real-time intraoperative imaging modality that enables in-depth perception of anatomical structures, guiding surgeons toward more precise tumor resections. As previously mentioned, the AR system is built by registering intraoperative and preoperative data, allowing more accurate surgical navigation. In this section, we will delve into the mathematical formulation of the registration problem, which is crucial to performing augmented reality, and provide a brief overview

of the state-of-the-art registration methods, with a particular focus on methods involving ultrasound.

1.5.1 The registration problem

Let M and F be the fixed and moving images, defined in domains $\Omega_M \subset \mathbb{R}^3$ and $\Omega_F \subset \mathbb{R}^3$, respectively. The domain Ω_M corresponds to the spatial region covered by the moving image, while Ω_F defines the spatial region of the fixed image. The registration problem aims at finding a transformation $\phi^* : \Omega_M \rightarrow \Omega_F$, that maximizes the similarity between features in M to homologous in F according to a given similarity criterion $\mathscr{L}_{sim}$. This transformation typically consists of two components. The first is a **rigid transformation**, denoted $T : \Omega_M \rightarrow \Omega_F$, which maps the domain Ω_M to Ω_F. This linear transformation includes a translation vector $\mathbf{t} \in \mathbb{R}^3$ and rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ transformations. For a point $x \in \Omega_M$, its rigid transformation is written as:

$$T(x) = \mathbf{t} + \mathbf{R}x \tag{1.1}$$

This rigid transformation handles global alignment but does not account for local deformations. A second component is introduced to address local variations and deformations: a non-linear or **non-rigid transformation** $u : \Omega_F \rightarrow \Omega_F$, which is applied after the rigid transformation. The non-rigid transformation is represented by a displacement field u that allows for more flexible alignment by deforming the moving image to match the fixed image more accurately, especially in cases where the organ or structure undergoes local shifts or deformations during surgery. Therefore, the combination of both transformations ϕ, for a given point $x \in \Omega_M$, reads as:

$$\phi(x) = T(x) + u(T(x)) \tag{1.2}$$

The optimal transformation ϕ^* is then found by maximizing a similarity metric that evaluates how well the transformed moving image aligns with the fixed image. This is mathematically formulated as:

$$\phi^* = \arg\max_{\phi} \mathscr{L}_{sim}(M \circ \phi, F) \tag{1.3}$$

Solving Equation 1.3 typically relies on optimization methods. Classical optimization techniques are typically employed to iteratively adjust the transformation parameters to minimize the difference between the two images. For rigid registration, gradient-based methods such as gradient descent [Baldi (1995)], Gauss-Newton [Pratt et al. (1998)], or

Levenberg-Marquardt [Moré (1978)] are commonly used. These methods compute the gradient of the similarity metric with respect to the transformation parameters and update the transformation iteratively until convergence. For non-rigid registration, where complex deformations are required, more advanced optimization methods are often applied. Variational approaches and regularization techniques [Henn et Witsch (2005)] are used to balance the trade-off between image similarity and transformation smoothness. For example, methods based on B-splines or thin-plate splines [Oliveira et Tavares (2014)] are used to model the deformation field, with optimization ensuring both accurate alignment and physically plausible deformations. Recently, Deep learning-based registration methods have gained prominence due to their ability to learn complex transformations and handle non-linear deformations more efficiently [Fu et al. (2020)]. These methods use convolutional neural networks (CNNs) to predict the transformation parameters directly from the image data. In supervised learning approaches, a neural network is trained to learn the mapping between image pairs and their corresponding transformation parameters [Eppenhof et Pluim (2019)] (e.g., translation, rotation, and deformation fields). The training process requires ground-truth transformations, which are often obtained from manually annotated datasets [Hu et al. (2018)]. In unsupervised deep learning approaches, the network learns to register images without explicit ground-truth transformations [Balakrishnan et al. (2018)]. These methods minimize a loss function that incorporates a similarity metric directly between the moving and fixed images and a regularization term that ensures smooth and realistic deformations. One popular framework is VoxelMorph, introduced by [Balakrishnan et al. (2019)], which uses a CNN to predict the displacement field that aligns two medical images.

The definition of the similarity metric is essential to ensure an accurate alignment between fixed and moving images. Two main approaches can be used to compute this similarity. The first approach is **intensity-based**, where the similarity is computed directly within the image space, comparing pixel or voxel intensities without requiring explicit segmentation. The second approach is **feature-based**. It involves detecting specific features or anatomical structures from both images and calculating the similarity based on those extracted features in Euclidean space. This method relies on explicit geometrical or structural features for alignment. These features can be specific points on the organ or internal structures surface extracted through segmentation process.

1.5.2 Intensity-based registration

In intensity-based registration methods, the similarity between the moving image M and the fixed image F is computed directly based on the intensity values at corresponding

pixels or voxels in the two images. This approach assumes that there is a meaningful relationship between the intensities of corresponding points in both images, and the registration is driven by this intensity relationship. Depending on the imaging modalities, intensity values may be directly comparable (e.g., in mono-modality registration) or may require more complex statistical relationships (e.g., in multi-modality registration). This leads to two distinct cases in intensity-based registration. In **mono-modal registration**, where the images come from the same modality (such as two US scans), Identity and Linear mappings are commonly considered [Roche et al. (2000)]. The identity mapping assumes that the moving and fixed images are identical once they are registered. The Sum of Squared Differences (SSD), commonly used as a similarity metric for mono-modal registration [Mohammadi et Keyvanpour (2021)], relies on this assumption. Introduced by Barnea et Silverman (1972), it is formulated as:

$$\mathrm{SSD}(F, M) = \frac{1}{N} \sum_{x \in \Omega_{F \cap M}} |M(x) - F(x)|^2 \tag{1.4}$$

Where $\Omega_{F \cap M}$ represents the overlap between the Ω_M and Ω_F. N is the number of points in the overlap. However, SSD is effective only when there are no significant changes in the contrast between the moving and fixed images [Hill et al. (2001)]. This limitation becomes particularly evident in imaging modalities such as ultrasound, where noise and intensity variations are prevalent. Ultrasound images, including speckle noise, can distort intensity values and reduce the effectiveness of SSD, making it difficult to achieve accurate alignment. To overcome this limitation, Rosenfeld (1976) proposed the Normalized Cross Correlation (NCC) similarity, which assumes the intensities of M as a linear function of the intensities of F when both are aligned, it reads as:

$$\mathrm{NCC}(F, M) = \frac{\sum_{x \in \Omega_{F \cap M}} (M(x) - \bar{M})(F(x) - \bar{F})}{\sqrt{\sum_{x \in \Omega_{F \cap M}} (M(x) - \bar{M})^2 \sum_{x \in \Omega_{F \cap M}} (F(x) - \bar{F})^2}} \tag{1.5}$$

In Equation 1.5, $\bar{M}$ and $\bar{F}$ are the mean intensities in the images M and F respectively. This metric measures how both image intensities are linked by a linear relation when aligned. Although NCC is more general than SSD and effective for mono-modal registration problems, numerous studies have shown that it is not sufficiently robust for multi-modal scenarios [Mohammadi et Keyvanpour (2021); Roche et al. (2000)].

For **mutli-modal** registration, the statistical dependency between the intensity values of the images is usually measured to evaluate the similarity between images [Tingting et Ning (2013)]. The Mutual Information (MI) is the most popular function for the purpose [Studholme et al. (1997)]. Introduced in the work of Collignon et al. (1995) and the works of Wells et al. (1996), MI quantifies how much knowing the intensity value of one image

reduces the uncertainty on the intensity value in the other image [Viola et Wells (1997)] and is expressed as:

$$\mathrm{MI}(F,M) = \sum_{x,y\in\Omega_{F\cap M}} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} \tag{1.6}$$

This equation is similar to the concept of Kullback-Leibler divergence, which is used to measure the difference between two probability distributions. In this context, MI(F,M) measures the statistical dependence between the two image distributions, F and M. Specifically, it compares the joint distribution $p(x,y)$, which represents the likelihood of intensity pairs occurring together in both images, to the product of the marginal distributions $p(x)p(y)$, which represents the scenario where the intensity values of F and M are statistically independent. When the joint distribution is similar to the product of the marginals, the images share little or no information, indicating independence. Conversely, higher MI signifies greater dependency between the two distributions, meaning the images share more structural or intensity-related information. This makes MI suitable for multi-modal registration, as it can capture the relationship between images with different intensity scales or contrasts, unlike SSD and NCC similarities presented earlier.

MI and its variants like Normalized Mutual Information (NMI) proposed by [Studholme et al. (1999)], and Entropy Correlation Coefficient (ECC) proposed by [Maes et al. (1997)] have been effective in various multi-modal registration problems. Nevertheless, many studies have shown that MI shows weaknesses in the context of US images [Roche et al. (2000); Woods et al. (1992)]. This limitation can be attributed to both the inherent intensity variations in US images and their unique physical imaging principles. Unlike other imaging modalities, which often display clear intensity patterns, ultrasound images primarily show contrast at the boundaries between different tissues. This is because ultrasound signals are reflected at interfaces with a change in acoustic impedance. Consequently, regions within the tissue interfaces, where intensity patterns are less defined and often obscured by speckle noise, do not provide consistent intensity information. As a result, successful registration of ultrasound images with other modalities must focus on capturing and aligning spatial information related to these tissue interfaces rather than relying solely on intensity-based similarities. Various MI variants have been developed and adapted to address this limitation [Rivaz et al. (2015)]. For instance, Roche et al. (2000) introduced a modified version of MI that incorporates functional mapping [Wein et al. (2007)] between the intensities of the fixed and moving images.

1.5.3 Feature-based registration

Feature-based registration is a technique used to align images by identifying and matching distinct features, such as points, edges, or landmarks, common between them [Abdel-Basset et al. (2017)]. This approach has been widely used in multi-modal medical imaging, where aligning images from different modalities (e.g., CT, MRI, US) is challenging due to their varying intensity profiles [Guan et al. (2018)]. As explained earlier, in multi-modal registration, voxel intensities may not directly correspond between modalities because each captures different aspects of anatomy. However, feature-based methods overcome this limitation by focusing on consistent anatomical landmarks, such as bones, organs, or tumors, which are present in both images. These features are typically obtained by segmenting the fixed and moving images, allowing the registration process to concentrate on aligning the key anatomical structures. The similarity in feature-based registration is usually measured by calculating the Euclidean distance between corresponding features in both images, with the aim of minimizing these distances for optimal alignment [Guan et al. (2018)].

The simplest feature-based method is Point-based Registration, originally proposed by Arun et al. (1987). In this technique, a set of corresponding points, known as fiducials, is selected from the two images being registered. The algorithm then applies a least-squares fitting process to minimize the distance between these matched points, resulting in a rigid transformation that aligns the two datasets. While this method is effective when dealing with small, well-defined sets of points, it becomes increasingly challenging as the size of the point sets grows. Larger datasets often lack distinct, easily identifiable features, making manual point matching difficult. To address this issue, Iterative Closest Point (ICP), introduced by Besl et McKay (1992) and later extended by Chen et Medioni (2002), automates the matching process between two sets of points. Instead of manually selecting corresponding points, ICP iteratively finds the closest points between the two point sets and applies transformations to align them. Two key variants of ICP have been developed: **point-to-point** ICP and **point-to-plane** ICP.

In **point-to-point** ICP [Besl et McKay (1992)], the algorithm minimizes the Euclidean distance between corresponding points from the two sets. For each point in the moving image, the closest point in the fixed image is identified, and a rigid transformation is computed to minimize the sum of squared distances between these matched points. On the other hand, **point-to-plane** ICP [Chen et Medioni (2002)] improves upon point-to-point ICP by incorporating the local geometry of the surface. Instead of minimizing the distance between corresponding points, point-to-plane ICP minimizes the distance between each point in the moving set and the plane defined by the surface normal at the correspond-

ing point in the fixed set. This approach leverages additional geometric information by considering the orientation of the surface. Both ICP variants are widely used in various applications, with the latter often preferred for datasets with smooth surfaces due to its stability. However, ICP is inherently limited to rigid transformations, which makes it inadequate for registering images with non-linear deformations, such as those involving soft tissues in medical imaging.

Non-rigid feature-based registration methods have been developed to address non-linear deformations. One of the simpler approaches involves estimating point-wise transformations, such as affine transformations, under smoothness regularization constraints [Liao et al. (2009)]. Other methods, like Coherent Point Drift (CPD) [Hirose (2021); Myronenko et Song (2010)], use a Gaussian Mixture Model (GMM) to build a 3D displacement field, ensuring coherent motion between nearby points. Another strategy is using deformation graphs, representing the scene as a sparse graph, and deformations are propagated from graph nodes to the entire surface. This method captures deformations while maintaining computational feasibility [Sumner et al. (2007)]. The Non-rigid Iterative Closest Point (NICP) algorithm further extends ICP by simultaneously optimizing the alignment and deformation regularized by a deformation graph [Li et al. (2008)]. Learning-based approaches, such as FlowNet3D [Dosovitskiy et al. (2015)], use 3D neural networks to predict displacement fields between point clouds, offering real-time solutions for non-rigid registration. Although these methods are fast, they can struggle with large deformations. More recent techniques, like Lepard [Li et al. (2022)], incorporate Transformers [Vaswani et al. (2017)] to improve point-to-point mapping to guide non-rigid registration. Spectral methods, such as functional map approaches [Ovsjanikov et al. (2012)], estimate correspondence in the spectral domain but are often limited to connected surfaces.

While these methods have advanced non-rigid registration considerably, **physics-based registration** techniques provide an even more anatomically accurate solution, particularly for medical imaging [Cotin et al. (1999); Courtecuisse et al. (2010b)]. These methods employ biomechanical models that simulate the physical properties of tissues, allowing for realistic deformation under breathing, surgical manipulation, or organ motion [Plantefeve et al. (2016); Suwelack et al. (2014)]. By incorporating biomechanical constraints that mimic how soft tissues behave in real-world scenarios, these physics-based approaches ensure that the registration aligns the images and respects the underlying physical and anatomical structure [Brunet (2020); Plantefeve et al. (2016)]. These biomechanical models often employ the principles of continuum mechanics governed by physical laws such as conservation of momentum, mass, and energy. To solve the resulting system of partial differential equations (PDEs), numerical methods like the finite element method (FEM) are typically used.

A common approach within these physics-based frameworks is using hyperelastic formulations, which are well-suited to simulate the behavior of biological tissues [Cotin et al. (1999)]. Hyperelastic materials can undergo large deformations, making them suitable to capture complex, nonlinear behavior of organs like the liver [Nikolaev et Cotin (2020)]. These formulations model the tissues' ability to stretch and compress in response to forces, showing how organs deform under normal physiological conditions and during medical interventions. A significant advantage of biomechanical models is their ability to extrapolate the deformation of the organ even when intraoperative data is incomplete or partial (e.g., Ultrasound). They can simulate the full behavior of the organ based on partial information. This is particularly useful when only limited intraoperative data is available, as the biomechanical model can predict the organ's deformation in areas where direct data might be missing while still adhering to realistic physical constraints [Brunet (2020); Plantefeve et al. (2016); Tagliabue et al. (2021)].

1.5.4 Intraoperative Ultrasound registration

Ultrasound registration has seen many works proposed, particularly in the context of liver surgeries where accurate real-time image alignment is essential due to the liver's soft, deformable nature. Depending on the clinical application, different types of ultrasound transducers, **LUS**, **standard** (external), or **IVUS**, are employed, each influencing the complexity and effectiveness of the registration methods.

For intensity-based methods, Banerjee et al. (2019) developed a block-matching framework for affine registration of 3D external ultrasound to CT. This approach, akin to a form of intensity-based Iterative Closest Point (ICP), samples both images into patches (blocks) and matches those with the highest intensity variance using a specific similarity measure. To handle the noisy correspondences between ultrasound and CT, they utilized a robust outlier rejection method based on game theory, previously applied in mono-modal ultrasound registration [Dagon et al. (2008); Nam et al. (2012)]. The Multiple Correlation Coefficient (MCC) was introduced as a similarity metric and used for block correspondence. The method was validated on 17 patients [Woods et al. (1992)]. Xu et al. (2013) used an ultrasound simulation model to guide the registration and compared simulated images with actual ultrasound using a combined intensity and gradient mutual information metric [He et Gu (2011)]. Their method achieved rigid registration between tracked freehand ultrasound and CT, validated with five clinical datasets. Wein et al. (2008, 2007) introduced the Linear Correlation of Linear Combination (LC^2) measure for affine registration of freehand ultrasound volumes to CT. Their approach incorporated intensity signals from both modalities and simulated ultrasound images derived from CT, which increased ro-

bustness. The LC^2 coefficients were calculated for small patches of the images, and the method was tested on 25 patients. In the case of LUS, most ultrasounds to CT registration methods face challenges due to the smaller sections of the liver captured, providing fewer data to constrain the registration process. While transabdominal ultrasound typically acquires complete liver sections [Penney et al. (2004); Wein et al. (2008)], LUS's limited field of view often requires manual selection of vessel bifurcations [Song et al. (2015)] or relies on a fairly accurate initial alignment to ensure proper registration [Ramalhinho et al. (2017)]. To address this limitation, Ramalhinho et al. (2019) introduced a globally optimal registration method that employs a Content-Based Image Retrieval (CBIR) approach to register untracked ultrasound slices. They create a discrete set of potential alignment solutions by simulating a finite number of ultrasound planes within the preoperative model.

Feature-based registration methods for ultrasound images of the liver primarily depend on the hepatic vascular trees [Aylward et al. (2003); Lange et al. (2003); Penney et al. (2001); Porter et al. (2001)] and the liver surface [Dehghan et al. (2015); Penney et al. (2001); Pohlman et al. (2019)]. For vessel surface features, Penney et al. (2001) proposed adapting the ICP algorithm that incorporated Gaussian noise to achieve rigid registration. Similarly, Porter et al. (2001) maximized the normalized cross-correlation (NCC) between segmented vessels from ultrasound and MRI images, while Aylward et al. (2003) focused on maximizing the overlap between ultrasound vessel positions and vessels from CT scans. However, all these approaches only address rigid registration. For non-rigid registration, Lange et al. (2003) developed an ICP method with an adapted correspondence step for vessel matching and Multi-level B-splines. Additionally, Dagon et al. (2008) introduced an elastic registration technique using mass-spring networks to model the soft tissue deformations. For liver surface features, Dehghan et al. (2015) proposed a method to maximize the overlap between the CT liver surface and the surface derived from an enhanced ultrasound volume map; their approach is rigid. Pohlman et al. (2019) utilized both vessel and liver surface features, employing a manual 2D plane setting as the initialization for point-based 2D affine registration.

The current state of the art in IVUS registration primarily focuses on its application in percutaneous coronary interventions (PCI), where methods aim to align IVUS images with angiographic data. This registration enhances the assessment of coronary artery disease (CAD) by enabling detailed visualization of the arterial wall while aiding in the delineation of culprit lesions, plaque characteristics, and identification of stent underexpansion. To cite a few, Hoffmann et al. (1999) highlighted the significance of assessing coronary artery morphology, indicating a strong link between plaque distribution and unstable angina. Sonka et al. (1995) implemented an intensity-based method to fuse biplane angiography with cross-sectional IVUS images, pointing out that the pullback path may

not always coincide with the lumen axis. Cothren et al. (2000) discussed how the accurate rotational alignment of the IVUS with the angiogram can be achieved through a "best-fit angle" function, with synchronization relying on temporal data.

1.6 IVUS-guided augmented reality

1.6.1 Thesis objective

While Urade et al. (2021) highlighted the benefits of using IVUS in computer-assisted interventions, as mentioned earlier in this chapter, no state-of-the-art research currently develops an IVUS registration method to assist in hepatic surgery. This thesis addresses this gap by developing an IVUS-guided AR system for minimally invasive liver surgery. The goal is to enhance surgical precision by integrating real-time IVUS imaging with AR technology, which overlays preoperative imaging data directly onto the surgical field. By leveraging the synchronous and continuous imaging capabilities of IVUS, which provides a comprehensive, full-thickness view of the liver intraoperatively through catheter rotation, the system would offer a detailed perspective of the organ's internal structures, including vascular trees and tumors. Such an AR is possible by registering intraoperative IVUS data with preoperative medical images such as CT or MRI. This registration process enables dynamic adjustments to preoperative models, ensuring they accurately reflect the liver's current anatomical state, thus addressing challenges such as organ deformation and limited visualization. To tackle the registration problem, we employ feature-based registration to circumvent issues related to multi-modality and the choice of similarity metrics. In addition, we utilize model-based registration by leveraging a biomechanical model of the liver built with preoperative data. This choice is motivated by the ability of such models to extrapolate effectively over sparse and noisy intraoperative data, like that obtained from IVUS. The system aims to facilitate more accurate tumor localization, improve navigation around vital structures, and optimize liver resections by equipping surgeons with an interactive and up-to-date view of the liver's internal anatomy. To the best of our knowledge, this is the first study to propose using IVUS as an intraoperative modality in the case of AR for liver surgery.

1.6.2 Contributions

1.6.2.1 First Author publications

1. **S. El hadramy**, A. Mazier, J. N .Brunet. J. S. Hale, S. Cotin, S. PA. Bordas. "SOniCS: Develop intuition on biomechanical systems through interactive error controlled

simulations". *Engineering with Computers* 40, 1857–1876 (2024)

2. **S. El hadramy**, A. Mazier, J. N .Brunet. J. S. Hale, S. Cotin, S. PA. Bordas. "SOniCS: Interfacing SOFA and FEniCS for advanced constitutive models". FEniCS Conference, San Diego, 2022.

3. **S. El hadramy**, J. Verde, N. Padoy, S. Cotin. In: Greenspan, H., et al. "Intraoperative CT augmentation for needle-based liver interventions". *Medical Image Computing and Computer Assisted Intervention – MICCAI 2023*. Lecture Notes in Computer Science, vol 14228.

4. **S. El hadramy**, J. Verde, KP. Beaudet, N. Padoy, S. Cotin. "Trackerless Volume Reconstruction from Intraoperative Ultrasound Images" In: Greenspan, H., et al. *Medical Image Computing and Computer Assisted Intervention – MICCAI 2023*. Lecture Notes in Computer Science, vol 14229.

5. **S. El hadramy**, J. Verde, N. Padoy, S. Cotin. "Towards Real-Time Vessel Guided Augmented Reality for Liver Surgery" *2024 IEEE International Symposium on Biomedical Imaging (ISBI)*, Athens, Greece, 2024, pp. 1-5.

6. **S. El hadramy**, N. Padoy, S. Cotin. "HyperU-Mesh: Real-time deformation of soft-tissues across variable patient-specific parameters". *To appear - Computational Biomechanics Workshop at MICCAI 2024*.

7. **S. El hadramy**, B. Acidi, N. Padoy, S. Cotin. "Optimization in latent space for real-time intraoperative characterization of digital twins". *Under review - Engineering with Computers*.

8. **S. El hadramy**, N. Padoy, S. Cotin. "Deform Any Liver In Real-time". *Under review - ISBI 2025*.

1.6.2.2 Other publications

9. KP. Beaudet, A. Karargyris, **S. El hadramy**, S. Cotin, JP. Mazier, N. Padoy, J. Verde. "Towards Real-time Intrahepatic Vessel Identification in Intraoperative Ultrasound-Guided Liver Surgery". To appear - *Medical Image Computing and Computer Assisted Intervention – MICCAI 2024*.

1.6.3 Communications

During my PhD program, I had the opportunity to present my research at several prestigious international conferences. I presented at MICCAI 2023 in Vancouver, MICCAI 2024

in Marrakesh, and ISBI 2024 in Athens. At MICCAI 2023, I gave two poster presentations at the main conference, while at ISBI 2024, I delivered an oral presentation. Additionally, I presented my work at the Computational Biomechanics Workshop at MICCAI 2024 with an oral talk. Furthermore, I was invited by Prof. Gabor Fichtinger (Queen's University) to serve as an instructor at the 3D Slicer tutorial during MICCAI 2024. The tutorial aims to help new, intermediate, and seasoned 3D Slicer users harness the various image segmentation and annotation tools effectively and, in doing so, open up the capabilities of AI-assisted medical image computing technologies to a wider audience. Furthermore, SOniCS, in chapter 4, was presented by my co-first author at the FEniCS 2022 Conference in San Diego USA, where it received an award. In addition to these international conferences, I have presented my work at the Mimesis Inria and the CAMMA IHU team meetings and seminars. Moreover, I have attended the Machine Learning for Healthcare Summer School at the University of Oxford, furthering my expertise in applying machine learning techniques to medical imaging. Finally, during the first two years of my PhD, I actively contributed to teaching at the Department of Mathematics of the University of Strasbourg.

1.7 Outlines

This thesis is organized according to the chapter structure outlined below:

- In **chapter 2**, we cover the fundamentals of deep learning and provide a detailed explanation of key architectures necessary for understanding the concepts presented in this thesis. This includes an overview of foundational techniques and specific models that will be referenced throughout the work.

- In **chapter 3**, we present the fundamentals of continuum mechanics for soft tissues. We focus on the relationship between stress and strain, explaining how these concepts are linked. Additionally, we demonstrate how to formulate a mechanical system and solve it using the finite element method (FEM). This foundation is essential for modeling the behavior of soft tissues in the context of this thesis.

- In **chapter 4**, we introduce the first contribution of this thesis: SOniCS, a framework designed to simplify the implementation of hyperelastic models. SOniCS integrates FEniCS, a tool for solving partial differential equations, with SOFA, a framework for real-time simulation of deformable objects. This integration allows for more efficient and user-friendly development of complex hyperelastic models, streamlining the simulation process for soft tissues and other applications.

- In **chapter 5**, we present our method for reconstructing an IVUS volume from a given sequence of frames. We present an AI-based method for volume reconstruction that eliminates the need for tracking, offering a flexible approach to reconstructing 3D IVUS volumes directly from the image sequences.
- In **chapter 6**, we present the key contribution of this thesis: a novel registration method that aligns preoperative CT data with intraoperative IVUS data. The intraoperative data is derived from the volume reconstructed using the methods discussed in Chapter 5. Meanwhile, the preoperative CT data is utilized to build a biomechanical model, which is crucial for the registration process. This model is developed using SOniCS, the framework introduced in Chapter 4.
- In **chapter 9**, we focus on enhancing the accuracy of the biomechanical model used for registration by proposing a novel method to identify patient-specific material properties and boundary conditions. This approach aims to refine the model's precision, ensuring a more accurate and personalized alignment between preoperative and intraoperative data.
- In **chapter 8**, we conclude this thesis by outlining the limitations of the techniques developed and discussing perspectives.

In appendix A, we present a work that is not directly related to the primary subject of this thesis but was developed during the early stages of this PhD program, when IVUS data was not yet available. This work contributed to our understanding of the state of the art in registration techniques and led to a publication at MICCAI 2023.

CHAPTER 2

DEEP NEURAL NETWORKS: FUNDAMENTALS

In this chapter, we lay the foundation of deep learning concepts necessary for understanding the techniques used throughout this thesis. We introduce the basic building blocks of neural networks, starting with the perceptron, which serves as the simplest form of a neural network. From there, we expand on this idea to discuss multilayer perceptrons (MLPs), which form the backbone of more complex neural architectures by connecting multiple layers of neurons to enable the network to model non-linear relationships in data. Once the basics are covered, we move on to explore more advanced architectures that are specifically employed in this thesis. First, we introduce Convolutional Neural Networks (CNNs), which are particularly powerful for analyzing structured grid-like data, such as images. Next, we discuss Long Short-Term Memory (LSTM) networks, a type of recurrent neural network (RNN) designed to handle sequential data and maintain long-term dependencies. We also introduce Convolutional LSTMs (ConvLSTMs), which combine the strengths of both CNNs and LSTMs, allowing for the modeling of spatial and temporal data simultaneously. We then cover Hypernetworks, an approach where one neural network generates the weights for another network. This architecture is used to dynamically adapt the parameters of the primary network based on context, allowing for flexibility and adaptability in the learning process. Finally, we explain the process of training deep neural networks. By establishing a solid understanding of these deep learning principles and methods, this chapter provides the theoretical grounding needed to comprehend and appreciate the advanced models and techniques applied in the later chapters of the thesis.

2.1 Introduction

In the late 1980s, neural networks emerged as a pivotal focus in Machine Learning (ML) and Artificial Intelligence (AI) domains. This surge in interest was largely driven by breakthroughs in developing more efficient learning algorithms and the design of sophisticated network architectures, which significantly enhanced the capabilities of these systems. A key milestone that further solidified the importance of neural networks was the development of the universal approximation theorem [Cybenko (1989), Hornik (1991)]. This theorem states that feedforward neural networks, composed of artificial neurons, can approximate any real-valued continuous function on compact subsets of $\mathbb{R}^n$. This theoretical foundation validated the potential of neural networks and paved the way for their widespread adoption in solving complex problems across various fields [Sarker (2021)]. As a result, neural networks quickly became a cornerstone of modern AI, laying the groundwork for the development of more advanced deep neural network techniques.

An artificial neuron, also known as a perceptron, is the basic unit of a neural network, designed to mimic the behavior of a biological neuron. As illustrated in Figure 2.1, an artificial neuron receives multiple inputs, each representing a feature from the data being processed. These inputs are assigned weights that indicate their significance in the neuron's calculations. The neuron computes a weighted sum of these inputs and adds a bias, an additional parameter added to the weighted sum, which allows the neuron to shift the activation function, providing greater flexibility in learning. This sum is then passed through an activation function, which determines the neuron's output by deciding whether it should be activated based on the input signals. The output of the neuron can either be used as input for other neurons in subsequent layers of the network or serve as the final output, depending on the architecture of the network. The interaction between these components enables the neuron to learn and adapt, making it a critical element in the functioning of neural networks.

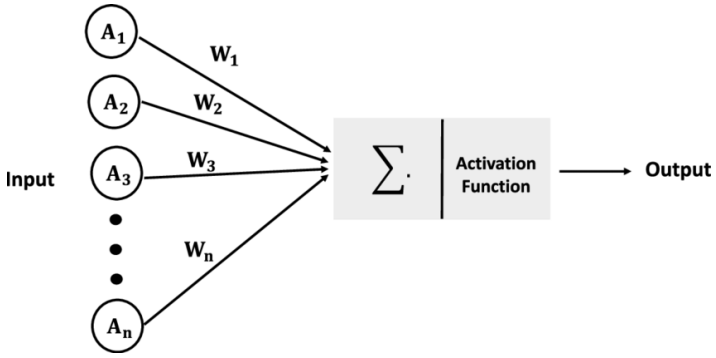


Figure 2.1: Illustration of an artificial neuron, where the input data is processed through a weighted sum, followed by the application of an activation function to introduce nonlinearity. This figure is adapted from Ghorbani et al. (2021).

In neural networks, the weighted sum of inputs plus a bias produces a linear combination of these inputs. Without activation functions, each neuron would compute a linear transformation of the input data, resulting in a model that can only learn linear relationships. However, many real-world problems involve complex patterns that cannot be captured by linear transformations alone. Activation functions add nonlinearity to the model by transforming the linear combination of inputs into a non-linear output. This nonlinearity allows the network to approximate a wide range of functions and capture intricate patterns within the data. Some popular activation functions are:

- **ReLU Function**: The Rectified Linear Unit (ReLU) function applies a threshold operation, where any negative input is set to zero, and positive input remains unchanged. $\text{ReLU}(x) = \max(0, x)$.

- **Sigmoid Function**: The sigmoid function maps the output to a range between 0 and 1, following the formula $\sigma(x) = \frac{1}{1+e^{-x}}$. Sigmoid introduces nonlinearity by compressing the output to a bounded range, which is useful for binary classification tasks.

- **The tanh function**: The tanh function, or hyperbolic tangent, maps the output to a range between -1 and 1, following the formula $\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$. This function introduces nonlinearity with a steeper curve than the sigmoid, and it centers the output around zero, which can help with convergence during training.

Other activation functions have been proposed in the literature. More details can be found in Dubey et al. (2022). Each activation function is designed to introduce nonlinearity in different ways and is tailored to specific problems and data structures. By incorporating these diverse activation functions, neural networks can capture complex, nonlinear relationships between inputs and outputs, thereby expanding their ability to address a wider array of challenges compared to models that rely solely on linear transformations.

2.2 Deep Neural Network Architectures

Various neural network architectures exist in the literature, each developed to address specific challenges and applications in different domains. For instance, Convolutional Neural Networks (CNNs) are widely used for image processing tasks, Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks are applied in sequential data analysis, such as time-series prediction or natural language processing, while Generative Adversarial Networks (GANs) are commonly used for tasks involving data generation. In this chapter, we focus on explaining the fundamental concepts of several architectures that are relevant to the work presented in this thesis. The list of architectures presented below is not exhaustive but serves as a foundation for the methods and models used in the following chapters. For further details, readers are invited to consult Schneider et Vlachos (2023).

2.2.1 Multi-Layer Perceptron

A Multi-Layer Perceptron (MLP) is the simpler architecture in artificial neural networks designed to model complex relationships within data through a series of interconnected layers of computational artificial neurons. The MLP architecture consists of three primary types of layers: an input layer, one or more hidden layers, and an output layer. Each layer serves a specific purpose: the input layer receives the raw data, the hidden layers process

and transform the data to capture underlying patterns, and the output layer produces the final predictions or classifications. Figure 2.2 illustrates an MLP architecture with an input layer of size 4, one hidden layer of size 5, and an output layer of size 1.

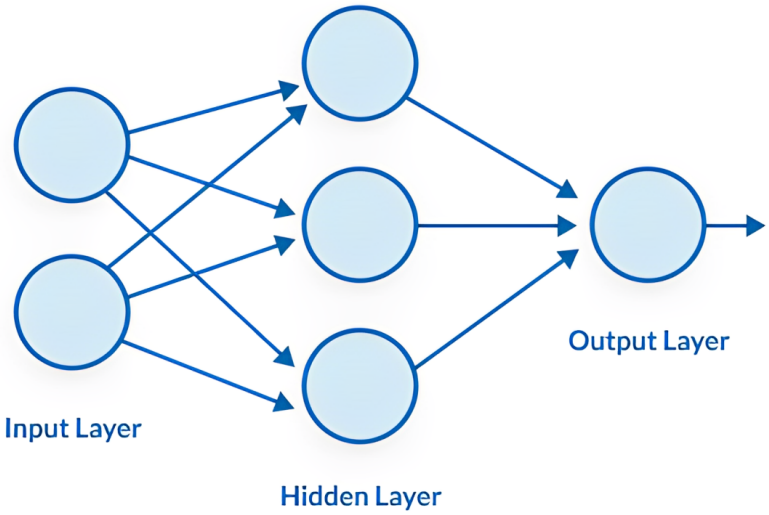


Figure 2.2: Illustration of a Multi-Layer Perceptron network, including an input layer of dimension 2, one hidden layer of dimension 3, and an output layer of dimension 1.In this architecture, each neuron in a layer is fully connected to every neuron in the preceding layer.

Within each layer, neurons are fully connected to every neuron in the adjacent layers, forming a densely interconnected network. The connections between neurons are associated with weights w_{ij}, which are crucial parameters that determine the strength of the signal passed from one neuron i in the previous layer to neuron j in the current layer. When data is fed into the network, each neuron calculates a weighted sum of its inputs following Equation 2.1.

$$z_j = \sum_{i=1}^{n} w_{ij} \cdot x_i + b_j \tag{2.1}$$

In Equation 2.1, z_j represents the weighted sum for neuron j, x_i is the input from neuron i, b_j is the bias term, and n is the number of inputs. This linear combination is then passed through an activation function $f(z_j)$ to introduce non-linearity following the Equation below.

$$a_j = f(z_j) \tag{2.2}$$

The output of the activation function a_j is then passed to the neurons in the next layer. This process of weighted summation, followed by non-linear activation, allows the MLP

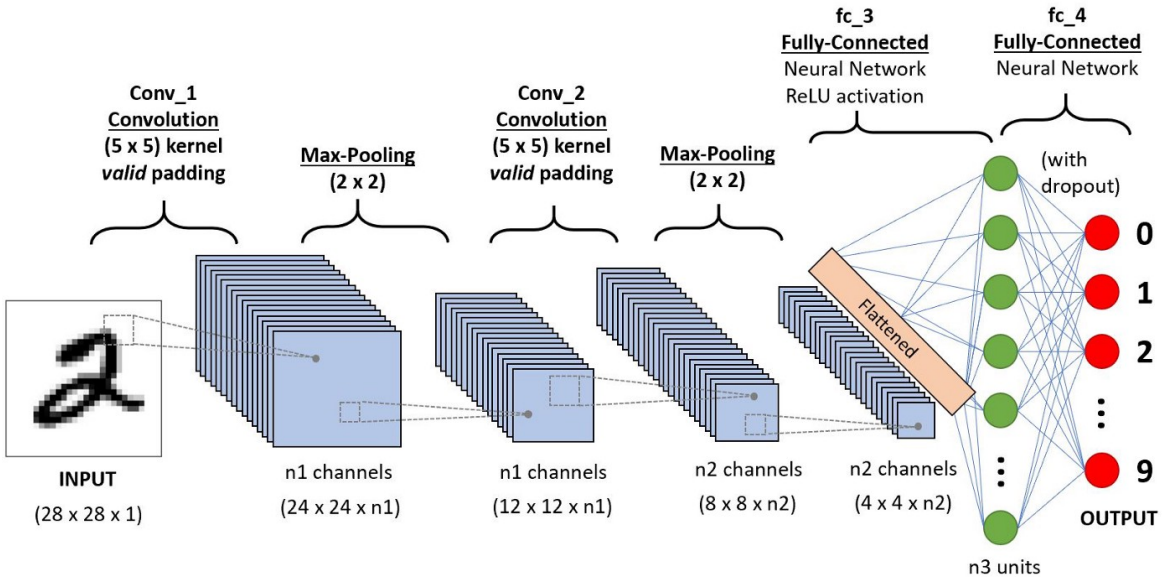


Figure 2.3: Example of a Convolutional Neural Network (CNN) incorporating convolutional, pooling, and fully connected layers. This particular network is designed to execute a classification task. The image is sourced from: www.towardsdatascience.com/

to model complex, non-linear relationships in the data. The final layer, the output layer, produces the network's predictions.

2.2.2 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a specialized deep learning architecture designed to effectively process and analyze structured grid-like data, such as images or videos. As illustrated in Figure 2.3, the architecture comprises several key components: convolutional layers, pooling layers, and fully connected layers, each contributing to the network's ability to learn spatial hierarchies of features.

2.2.2.1 Convolutional layers

The convolutional layer is the core component of a CNN and is responsible for detecting local patterns in the input data. It employs a set of learnable filters (or kernels), which are matrices that slide over the input data, performing the convolution operation. This operation involves calculating a weighted sum of the input values within the filter's receptive field and producing a feature map highlighting specific patterns in the data. For 2D data, the convolution operation at a specific location (i, j, k) in the feature map is given by:

$$\boldsymbol{Z_{ij}} = (\boldsymbol{X} * \boldsymbol{W})_{\boldsymbol{ij}} = \sum_{m=1}^{k} \sum_{n=1}^{k} \boldsymbol{X_{i+m-1,j+m-1}} \cdot \boldsymbol{W_{mn}} + \boldsymbol{b} \tag{2.3}$$

In Equation 2.3, $\boldsymbol{X}$ is the input data expressed as a matrix, $\boldsymbol{W}$ is the filter (also known as the kernel) matrix of size $k \times k$, $\boldsymbol{Z_{ij}}$ is the resulting value in the feature map at position (i, j), and $\boldsymbol{b}$ is the bias term. The filter $\boldsymbol{W}$ slides over the input data, covering different regions of size $k \times k$. The sliding window process is known as striding. If the stride is greater than 1, the filter moves by multiple pixels at a time, reducing the size of the output feature map. For 3D data, the convolution operation at a location (i, j, k) in the feature map is written as:

$$\boldsymbol{Z_{ijk}} = \sum_{d=1}^{D} \sum_{m=1}^{k_h} \sum_{n=1}^{k_w} \boldsymbol{X_{i+m-1,j+n-1,d}} \cdot \boldsymbol{W_{mnd}} + \boldsymbol{b} \tag{2.4}$$

In Equation 2.4, $\boldsymbol{X}$ is the input data of dimension 3, with D being the size of its third dimension, $\boldsymbol{W}$ is the 3D filter with dimensions $k_h \times k_w \times D$, $\boldsymbol{Z_{ijk}}$ is the feature map value at position (i, j, k), and k_h and k_w are the height and width of the filter. After each convolution layer, the resulting feature map is passed through an activation function to introduce non-linearity. The most commonly used activation function is the Rectified Linear Unit (ReLU), as explained previously.

2.2.2.2 Pooling layers

Pooling layers play a crucial role in CNNs by reducing the spatial dimensions of the feature maps generated by convolutional layers. This dimensionality reduction serves two primary purposes: it decreases the network's computational complexity, makes it more efficient, and enhances the model's robustness by making the learned features more invariant to small translations or distortions in the input data. Pooling operates over local regions of the input feature maps, similar to convolutional operations, but instead of applying a filter to detect patterns, it performs a down-sampling operation. The most common types of pooling are max pooling and average pooling.

- **Max pooling**: selects the maximum value within each region, capturing the most prominent feature within the region. For a $p \times p$ pooling region, the operation can be expressed as:

$$\boldsymbol{P_{ij}} = \max_{(m,n) \in p \times p} \boldsymbol{A}_{i+m-1,j+n-1} \tag{2.5}$$

- **Average pooling**: computes the average of the values within the pooling region,

providing a smoothed representation of the features, and is expressed as:

$$\boldsymbol{P}_{ij} = \frac{1}{p \times p} \sum_{m=1}^{p} \sum_{n=1}^{p} \boldsymbol{A}_{i+m-1, j+n-1} \tag{2.6}$$

In Equations 2.5 and 2.6, $\boldsymbol{P}_{ij}$ represents the pooled feature map value at position (i, j), and $\boldsymbol{A}$ is the input feature map. Pooling layers contribute to the network's ability to generalize by reducing sensitivity to the exact location of features, enabling the CNN to extract patterns regardless of small shifts or variations in the input. By progressively downsampling the feature maps, pooling layers also help manage the complexity of the network, allowing deeper architectures to be trained more efficiently.

2.2.2.3 Fully connected layers

After passing through multiple convolutional and pooling layers, the feature maps of these layers generate increasingly abstract and high-level representations of the input data. These feature maps, which still retain their spatial structure, must be converted into a format suitable for the final classification or regression task. To achieve this, the multi-dimensional feature maps are flattened into a one-dimensional vector. Flattening essentially unrolls the 2D or 3D feature maps into a single long vector, where each element of the vector corresponds to a specific feature detected in the previous layers. This vector is passed through fully connected layers, similar to the MLP presented earlier, where each neuron is connected to each neuron in the previous layer. These layers perform the final classification or regression tasks.

2.2.3 Long-Short-Term Memory Networks

Long-short-term memory (LSTM) networks are a specific type of neural network designed to capture long-range dependencies within sequential data. LSTMs are particularly effective in tasks where understanding temporal context is crucial, such as time series prediction, natural language processing, and speech recognition. The LSTM architecture, illustrated in Figure 2.4, incorporates a complex unit structure with three primary components: the cell state, the input gate, the forget gate, and the output gate. These components manage and retain information over extended sequences. An LSTM unit processes three vectors at each time step. Two of these vectors, the cell state $\boldsymbol{C_{t-1}}$ and the hidden state $\boldsymbol{H_{t-1}}$, are produced by the LSTM from the previous time step $t-1$. The third vector is the input vector $\boldsymbol{X_t}$, which is provided to the LSTM at the current time step t.

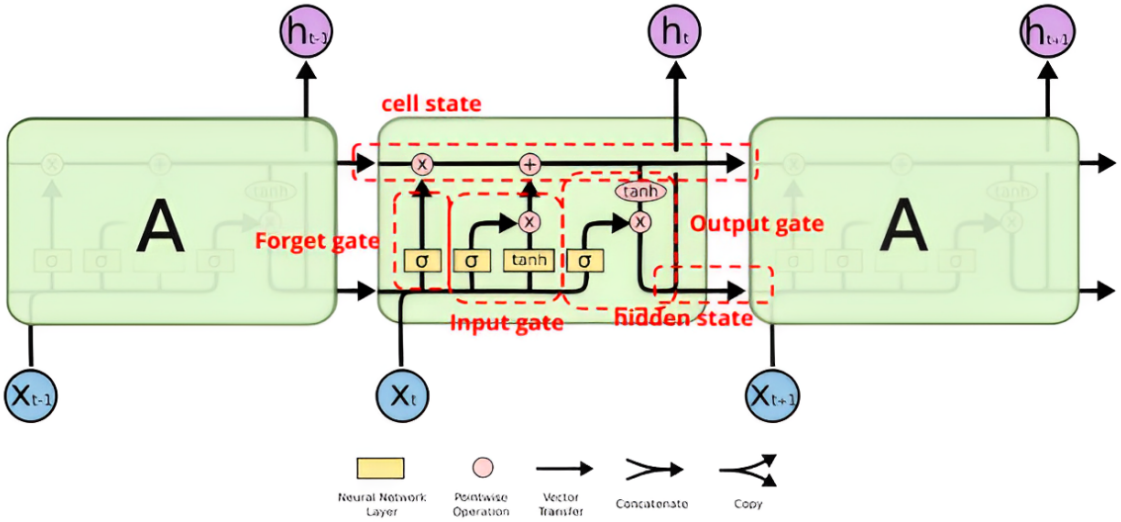


Figure 2.4: Illustration of a Long Short-Term Memory (LSTM) network, where each LSTM layer includes a cell state, hidden states, and three types of gates: input, forget, and output.

2.2.3.1 Cell state

The cell state is a central element of an LSTM unit that carries information across the sequence. It acts as a memory buffer, allowing information to flow through the network without significant changes. The cell state C_t at time step t is updated according to the following Equation:

$$C_t = f_t \otimes C_{t-1} + i_t \otimes \tilde{C}_t \tag{2.7}$$

In Equation 2.7, f_t is the forget gate's output, C_{t-1} is the cell state from the previous time step, i_t is the input gate's output, and $\tilde{C}_t$ is the candidate cell state generated by the network. Note that $\otimes$ is the point-wise multiplication.

2.2.3.2 Forget Gate

The forget gate decides which information from the cell state should be discarded. This is computed using a sigmoid activation function, producing values between 0 and 1:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.8}$$

Where in Equation 2.8, W_f is the weight matrix for the forget gate, h_{t-1} is the hidden state from the previous time step, x_t is the current input, b_f is the bias term, and σ denotes the sigmoid function.

2.2.3.3 Input Gate

The input gate controls the incorporation of new information into the cell state. It consists of two parts: a sigmoid layer that determines which values to update and a tanh layer that generates candidate values following Equations.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.9}$$

$$\tilde{C}_i = \tanh(W_C \cdot [h_{t-1}, x_t] + b_c) \tag{2.10}$$

In Equations 2.9 and 2.10, W_i and W_c are weight matrices for the input gate and candidate cell state, respectively. b_i and b_c are the respective biases. tanh denotes the hyperbolic tangent function.

2.2.3.4 Output Gate

The output gate determines the value of the hidden state outputted by the LSTM (in instant t) and received by the LSTM in the next instant $(t+1)$ input set. The hidden state h_t is computed following Equation 2.11.

$$h_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \otimes \tanh(C_t) \tag{2.11}$$

Where W_{t-1} and b_o are the weight and bias matrices of the output gate, respectively. LSTM networks are adept at capturing long-term dependencies in sequential data by maintaining and updating cell states across extended time steps. Additionally, LSTMs address the vanishing gradient problem [Pascanu et al. (2012)] through their gating mechanisms, which regulate the flow of gradients during training and enable effective learning over long sequences. By employing forget, input, and output gates, LSTMs manage memory efficiently, allowing them to retain critical information while discarding less relevant details.

2.2.4 Convolutional LSTM

A Convolutional Long Short-Term Memory (ConvLSTM) network (see Figure 2.5), introduced by Shi *et al.* [Shi et al. (2015)], integrates convolutional operations with LSTM units to capture spatial and temporal dependencies in sequential data. Unlike standard LSTM networks operating on flat vectors, ConvLSTM applies convolutional layers within the LSTM framework, allowing it to process and learn from spatial structures in data such as images or video frames. In a ConvLSTM unit, the input data, which consists of mul-

tiple frames or structured data, is processed through convolutional gates instead of fully connected layers. This means that the forget, input, and output gates and the cell state updates are performed using convolutional filters. Specifically, the input, cell state, and hidden state are all represented as multi-dimensional matrices, and convolutional operations are used to update these terms. This approach enables ConvLSTM to model complex spatial patterns and dynamic changes over time, making it suitable for tasks such as video prediction, spatiotemporal forecasting, and other applications where both spatial and temporal information are crucial.

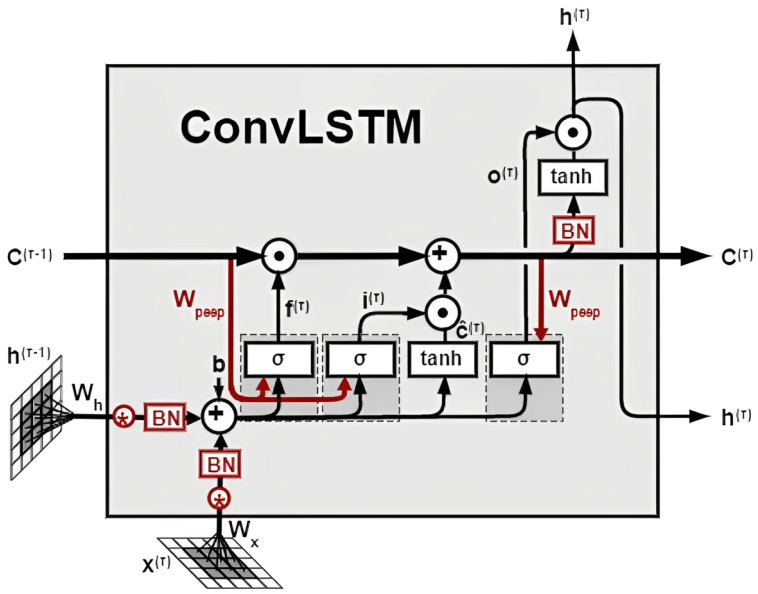


Figure 2.5: Illustration of a ConvLSTM layer, which retains the core structure of a traditional LSTM layer while incorporating convolutional operations. This allows the ConvLSTM to learn spatial features from the data and capture temporal dependencies. The Figure is sourced from: https://medium.com

2.2.5 HyperNetworks

Introduced by Ha *et al.* [Ha et al. (2017)], hypernetworks are a method where a network, referred to as the hypernetwork, generates the weights for another network, known as the primary network. The hypernetwork is trained to produce weights for the primary network. Instead of learning a fixed set of weights, the hypernetwork outputs weight matrices for each primary network layer based on its input. This can be represented as:

$$\boldsymbol{W}_i = \boldsymbol{h}(\boldsymbol{z}_i) \tag{2.12}$$

In Equation 2.12, $\boldsymbol{W}_i$ denotes the weight matrix for the i-th layer of the primary network. $\boldsymbol{h}$ is the hypernetwork that generates these weights, and $\boldsymbol{z}_i$ is the input for the hypernet-

work, which usually represents a conditioning parameter. The primary network utilizes the weights generated by the hypernetwork to perform predictions. For a given input $\boldsymbol{x}$, the prediction of the target network can be expressed as:

$$\hat{\boldsymbol{y}} = \boldsymbol{f}_{\boldsymbol{W}_i}(x) \tag{2.13}$$

With $\hat{\boldsymbol{y}}$ being the output of the primary network, $\boldsymbol{f}$ represents the architecture of the primary network, and $\boldsymbol{W}_i$ are the weights of the primary network, which are predicted by the hypernetwork $\boldsymbol{h}$. Hypernetworks provide a way for generating adaptive and context-sensitive parameters for neural networks, enhancing their flexibility and ability to generalize across different tasks.

2.3 Training Deep Neural Networks

Training a neural network involves optimizing its parameters to minimize the difference between predicted outputs and target values. The process begins with initializing the network's weights, typically using Gaussian distributions with small variance to ensure that the weights start from a small and diverse range of values [Narkhede et al. (2022)]. The network performs a forward pass for each training iteration to compute the predicted output $\boldsymbol{y}$ for a given input $\boldsymbol{x}$. This prediction is compared to the true target $\hat{\boldsymbol{y}}$ using a loss function $\mathscr{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})$, which quantifies the prediction error. The choice of loss function depends on the problem's nature and the data's structure. For instance, Mean Squared Error (MSE) is commonly employed for regression task, which is expressed as:

$$\mathscr{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{N} \sum_{i=1}^{N} ||\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i)||_2^2 \tag{2.14}$$

The loss function may vary for classification tasks based on the specific problem. For binary classification, Binary Cross-Entropy Loss is often used:

$$\mathscr{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\frac{1}{N} \sum_{i=1}^{N} [\hat{\boldsymbol{y}}_i \log(\boldsymbol{y}_i) + (1 - \hat{\boldsymbol{y}}_i) \log(1 - \boldsymbol{y}_i)] \tag{2.15}$$

In equations 2.14 and 2.15, N is the dimension of the output $\boldsymbol{y}$. Other loss functions are presented in the literature, each designed for specific types of problems and data. These include both supervised and unsupervised loss functions [Terven et al. (2023)]. In the context of medical imaging, particularly for segmentation tasks, the Dice Loss is frequently employed to evaluate the overlap between predicted and ground truth segmentations:

$$\text{Dice Loss} = 1 - \frac{2 \cdot |\text{A} \cap \text{B}|}{|\text{A} + \text{B}|} \tag{2.16}$$

Where A and B are the predicted and actual segmentation masks, respectively.

After the loss computation, backpropagation is used to compute the gradient of the loss function with respect to each weight $\boldsymbol{w}$ in the network. This gradient is computed using the chain rule and can be expressed as:

$$\frac{\partial \mathscr{L}}{\partial \boldsymbol{w_k}} = \frac{\partial \mathscr{L}}{\partial \boldsymbol{y}} \cdot \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{w_k}} \tag{2.17}$$

These gradients are used to find the weight adjustments that best minimize the loss function. An optimization algorithm, such as Stochastic Gradient Descent (SGD), updates the weights w following:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \frac{\partial \mathscr{L}}{\partial \boldsymbol{w}} \tag{2.18}$$

Where η is the learning rate.

This Training iterative process continues over multiple epochs, with each epoch involving passing the entire training dataset through the network. During training, techniques such as learning rate scheduling, regularization (e.g., L2 regularization), and dropout [Srivastava et al. (2014)] may be employed to enhance the network's performance and prevent overfitting.

2.4 Conclusion

This chapter has provided an overview of the fundamental concepts of deep learning, laying the groundwork for understanding the core principles that underpin the neural network architectures used throughout this thesis. We introduced the essential building blocks of deep learning, including the general structure of neural networks and their training process. We explored several specialized architectures, such as CNNs and LSTMs, emphasizing their suitability in structured and sequential data feature extraction. Particular attention was given to the specific architectures that form the backbone of the methods developed in this thesis. These architectures were selected based on their ability to address the challenges posed by the research problems tackled in later chapters. By establishing a theoretical foundation and focusing on the architectures that play a pivotal role in our work, this chapter sets the stage for the detailed exploration of novel methodologies and experiments that follow.

CHAPTER 3

SOFT-TISSUES COMPUTATIONAL BIOMECHANICS

This chapter presents the fundamentals of soft-tissue computational biomechanics. It begins with introducing the fundamental principles of continuum mechanics, focusing on key concepts such as the representation of internal forces through the stress tensor and the measurement of deformation using strain tensors. The chapter also outlines the constitutive laws that relate these quantities and formulates the central problem to be addressed. Following this, the finite element method (FEM) is introduced as a powerful computational tool for solving complex biomechanical problems. The discussion covers the discretization of the domain, the construction of shape functions, the assembly of the system, and the numerical methods used to solve it. This chapter provides the requirements for understanding soft tissue simulation in the context of the thesis. Most of the concepts and mathematical equations presented in this chapter are inspired by Wriggers (2008).

3.1 Continuum mechanics

Continuum mechanics is a branch of mechanics that describes the behavior of materials modeled as continuous rather than discrete entities. It provides a way to analyze how materials deform and respond to internal and external forces. Internal forces are crucial for maintaining a body's structural integrity, preventing it from being stretched or compressed indefinitely. When forces are applied to a deformable body, they induce internal constraints, known as stress, and result in deformations, referred to as strain. These concepts are examined in detail in the subsequent sections. We first introduce the deformation gradient and then define and analyze stress and strain, exploring how they are quantified and their roles in describing the internal behavior of materials under external forces. Following this, we discuss how these concepts are applied in the formulation of physical models and how they contribute to predicting a material's response to various loading conditions. Through this exploration, we develop a comprehensive understanding of how internal forces and deformations interact to shape the behavior of deformable bodies.

3.1.1 The Deformation Gradient

The deformation gradient is a fundamental tensor in continuum mechanics that describes the local deformation of a material. It captures how a small region of a material body is transformed from its initial (reference) configuration to its deformed (current) configuration. The equations that describe deformation can be expressed using the *Lagrangian* or *Eulerian* descriptions, each offering a different perspective on the material's behav-

ior. The *Lagrangian* description, also known as the material description, tracks individual particles of the material as they move from their original (reference) configuration to their deformed state. This approach focuses on the initial position of each particle, making it particularly useful for studying how specific points in the material deform over time. On the other hand, the *Eulerian* description, or spatial description, examines the deformation from the perspective of a fixed spatial coordinate system. This approach focuses on specific locations in space, observing how material flows through these points as it deforms.

Given $\mathbb{E}^d$, a space of dimension d, let $E = \{\mathbf{E_i} \in \mathbb{E}^d\}$ with $i = \{1, ..., d\}$ be the orthonormal basis vectors of an Euclidean vector in the space $\mathbb{E}^d$ with origin O. The object of interest is a subset domain $\Omega \subset \mathbb{E}^d$. This domain can be seen as a region within the object's boundary. By incorporating the concept of time, we define the state of rest as the object's shape at time $t = t_0$, representing the object's initial configuration before any deformation begins. The applied forces will deform the object throughout the simulation, causing the domain Ω to vary over time. The position vector of a point within the initial domain is denoted as $\boldsymbol{X} \in \Omega_0$, with $\Omega_0 = \Omega(t_0)$. The coordinate of $\boldsymbol{X}$, known as material coordinates, serves as the independent variable of the system. To determine the position vector of a material point in the deformed domain at a given time t, noted $\boldsymbol{x_t}$, we define the linear map $\boldsymbol{\phi_t} : \Omega_0 \rightarrow \Omega_t$ as a rigid transformation between the initial and deformed domains Ω_0 and Ω_t respectively. This rigid transformation is represented as a rotation $\boldsymbol{R_t}$ and translation $\boldsymbol{T_t}$ components. The position vector of a deformed material point reads as:

$$\boldsymbol{x_t} = \boldsymbol{\phi_t}(\boldsymbol{X}) = \boldsymbol{R_t} \cdot \boldsymbol{X} + \boldsymbol{T_t} \tag{3.1}$$

The displacement vector of a point from the domain Ω_0 to Ω_t is denoted by $\boldsymbol{u_t}$ and is expressed as:

$$\boldsymbol{u_t}(\boldsymbol{X}) = \boldsymbol{x_t} - \boldsymbol{X} = \boldsymbol{\phi_t}(\boldsymbol{X}) - \boldsymbol{X} \tag{3.2}$$

From this point forward, the subscript t will be omitted, with the understanding that $\boldsymbol{x}$, $\boldsymbol{\phi}$ and $\boldsymbol{u}$ are evaluated at a specific time t. The *deformation gradient* is defined as the partial derivative of the transformation $\boldsymbol{\phi}$ with respect to the material coordinate and is expressed as:

$$\mathbb{F} = \boldsymbol{\nabla}_{\boldsymbol{X}}[\boldsymbol{\phi}(\boldsymbol{X})] = \boldsymbol{\nabla}_{\boldsymbol{X}}[\boldsymbol{X} + \boldsymbol{u}] = \mathbb{I}_d + \boldsymbol{\nabla}_{\boldsymbol{X}}\boldsymbol{u} \tag{3.3}$$

Where $\boldsymbol{\nabla}_{\boldsymbol{X}}\boldsymbol{u}$ is the *displacement gradient tensor*, the *deformation gradient* $\mathbb{F}$ provides a comprehensive measure of how a material deforms from its original configuration to its

current state. $\mathbb{I}_d$ represents the identity matrix of dimension d. $\mathbb{F}$ is a fundamental tensor that describes the local transformation of infinitesimal material elements due to applied forces or boundary conditions. It relates tangent vectors of initial and current domains to each other. For instance, it maps an infinitesimal segment from the initial $(\boldsymbol{X}, \boldsymbol{X}+\boldsymbol{dX})$ to $(\boldsymbol{x}, \boldsymbol{x}+\boldsymbol{dx})$ following the relation:

$$\boldsymbol{dx} = \mathbb{F}\boldsymbol{dX} \tag{3.4}$$

In the same manner, it maps a small area $\boldsymbol{dA}$ with a unit normal vector $\boldsymbol{N}$ to its corresponding area $\boldsymbol{da}$ with a unit vector $\boldsymbol{n}$. And a small volume element $\boldsymbol{dV}$ to its deformed shape $\boldsymbol{dv}$ as follows:

$$\boldsymbol{dan} = \boldsymbol{J}\boldsymbol{dA}\mathbb{F}^{-\boldsymbol{T}}\boldsymbol{N} \tag{3.5}$$

$$\boldsymbol{dv} = \boldsymbol{J}\boldsymbol{dV} \tag{3.6}$$

In Equations 3.5 and 3.6, $\boldsymbol{J}$ is the determinant of $\mathbb{F}$ and indicates volume preservation, which is influenced by the material's compressibility. For an incompressible material, $\boldsymbol{J}$ must equal 1.

3.1.2 Definition of the stress tensor

Stress is a physical quantity that represents the internal distribution of force per unit area within a material, resulting from external loads, body forces, or changes in temperature. Let $\boldsymbol{dS}$ be an infinitesimal area with normal $\boldsymbol{n}$ in a given volume $\boldsymbol{V}$. at a point $\boldsymbol{x}$ of the volume inside the $\boldsymbol{dS}$. The stress vector $\boldsymbol{\sigma_n}$ is defined as the average force per unit area and is expressed as:

$$\boldsymbol{\sigma_n} = \frac{\boldsymbol{f}}{\boldsymbol{dS}} \tag{3.7}$$

In Equation 3.7, $\boldsymbol{f}$ is the force applied on the infinitesimal surface $\boldsymbol{dS}$. This stress vector depends on the normal of $\boldsymbol{dS}$ and thus only characterizes the stress on that particular surface. However, this definition is insufficient to fully describe the stress at a point within the volume. To define the stress at a point $\boldsymbol{x}$, we need to consider the stress tensor, which represents the stresses acting on surfaces of all possible orientations passing through that point. Consider an infinitesimal tetrahedral volume element $\boldsymbol{dV}$ with outward-pointing normal vectors on its surfaces, as illustrated in Fig 3.1. The force acting on the larger face with a normal vector $\boldsymbol{n}$ is given by:

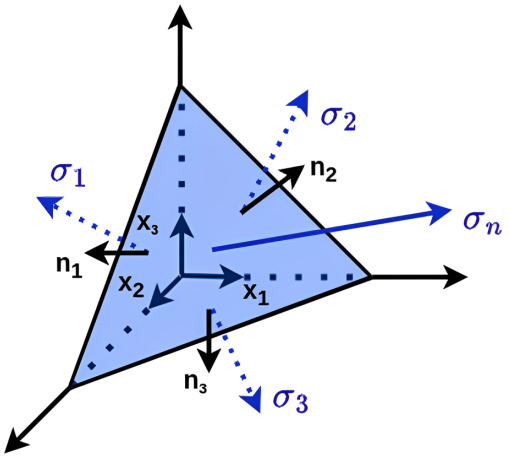


Figure 3.1: An infinitesimal tetrahedral volume element illustrating the stress vectors $\boldsymbol{\sigma_1}$, $\boldsymbol{\sigma_2}$, $\boldsymbol{\sigma_3}$ and $\boldsymbol{\sigma_n}$, associated with the corresponding normal vector $\boldsymbol{n_1} = -\boldsymbol{x_1}$, $\boldsymbol{n_2} = -\boldsymbol{x_2}$, $\boldsymbol{n_3} = -\boldsymbol{x_3}$ and $\boldsymbol{n}$, respectively. These stress vectors represent the internal forces per unit area acting on the faces of the tetrahedron.

$$\boldsymbol{f} = \boldsymbol{\sigma_n} \boldsymbol{dS} \tag{3.8}$$

In the Equation 3.8, dS is the area of the considered face. For each face of the tetrahedron, the force is expressed following the Equation 3.9.

$$\boldsymbol{f_i} = -\boldsymbol{\sigma_i} \boldsymbol{dS_i} \tag{3.9}$$

With $i \in \{1,2,3\}$ and $\boldsymbol{\sigma_i}$ is the stress vector on the i^{th} face of the tetrahedron noted $\boldsymbol{dS_i}$. We apply the principle of conservation of linear momentum to the infinitesimal volume $\boldsymbol{dV}$ of the tetrahedron, considering the volume force inside the tetrahedron, noted $\boldsymbol{f_v dV}$:

$$\boldsymbol{f} + \boldsymbol{f_1} + \boldsymbol{f_2} + \boldsymbol{f_3} + \boldsymbol{f_v} \boldsymbol{dV} = \rho \boldsymbol{dV} \frac{\boldsymbol{dv}}{\boldsymbol{dt}} \tag{3.10}$$

Here, ρ is the mass density, and v is the velocity of the tetrahedron. To further develop this, we express dV as $d \cdot h \cdot c$, where h is the height of the tetrahedron and c is a constant depending on h. We also express the forces with the stress vectors following Equation 3.9. Substituting this into Equation 3.10 gives:

$$\boldsymbol{\sigma_n} dS - \boldsymbol{t_1} n_1 dS - \boldsymbol{t_2} n_2 dS - \boldsymbol{t_3} n_3 dS + \boldsymbol{f_v} hcdS = \rho hcdS \frac{dv}{dt} \tag{3.11}$$

Devinding by dS in Equation 3.11 and tending h to zero leads to:

$$\boldsymbol{t_n} = \begin{pmatrix} \boldsymbol{t_1} & \boldsymbol{t_2} & \boldsymbol{t_3} \end{pmatrix} \cdot \begin{pmatrix} n_1 \\ n_2 \\ n_3 \end{pmatrix} = \begin{pmatrix} \boldsymbol{\sigma_{11}} & \boldsymbol{\sigma_{12}} & \boldsymbol{\sigma_{13}} \\ \boldsymbol{\sigma_{21}} & \boldsymbol{\sigma_{22}} & \boldsymbol{\sigma_{23}} \\ \boldsymbol{\sigma_{31}} & \boldsymbol{\sigma_{32}} & \boldsymbol{\sigma_{33}} \end{pmatrix} \cdot \begin{pmatrix} n_1 \\ n_2 \\ n_3 \end{pmatrix} = \boldsymbol{\sigma_c} \cdot n \tag{3.12}$$

This means that when the stress vectors on the tree faces of the tetrahedron are known, we can compute the stress vector for any normal direction n. $\boldsymbol{\sigma_c}$ denotes the Cauchy stress tensor, which measures the intensity of the internal forces acting on a material per unit area in its **deformed** configuration. The second Piola-Kirchhoff tensor $\boldsymbol{\sigma_p}$ is another way to define the stress. Unlike the Cauchy stress tensor, the Piola Kirchhoff tensor provides a measure of the internal forces per unit area in the **undeformed** configuration. It reads as:

$$\boldsymbol{\sigma_s} = \boldsymbol{J}\mathbb{F}^{-1} \cdot \boldsymbol{\sigma_c} \cdot \mathbb{F}^{-T} \tag{3.13}$$

Where $\mathbb{F}$ is the deformation gradient defined in the previous section and $\boldsymbol{J} = \det(\mathbb{F})$ is the jacobian matrix allowing to pass from the rest to the deformed configuration.

3.1.3 Definition of the strain tensor

In the previous section, we discussed how the forces applied to a material affect its internal structure through the stress tensor. We will now explore the mathematical definition of deformation by introducing a concept known as strain, which quantifies this deformation. Strain quantifies how much a material deforms relative to its original configuration when subjected to forces.

Considering a continuum body that occupies a domain Ω of boundary Γ. Let $\boldsymbol{X}$ and $\boldsymbol{X} + \boldsymbol{dX}$ be two points infinitesimally close in the rest configuration, $\boldsymbol{x}$ and $\boldsymbol{x} + \boldsymbol{dx}$, their respective position after a given deformation of the solid. As seen in the section 3.1.2, the segments $(\boldsymbol{X}, \boldsymbol{X} + \boldsymbol{dX})$ and $(\boldsymbol{x}, \boldsymbol{x} + \boldsymbol{dx})$ are related with the *deformation gradient* with the relation $\boldsymbol{dx} = \mathbb{F}\boldsymbol{dX}$. With $\mathbb{F} := \boldsymbol{\nabla u} + \mathbb{I}$ being the deformation gradient and $\boldsymbol{u}$ the displacement field. The gradient of the displacement field, $\boldsymbol{\nabla u}$, describes the relationship between the segments connecting points in both the reference and deformed configurations. When $\boldsymbol{\nabla u} = 0$, it indicates that the material has undergone only a rigid translation with no deformation. However, $\boldsymbol{\nabla u}$ is non-zero when the body experiences a rotation, even in the absence of deformation. Therefore, to fully capture the deformation of the material, measuring the change in angle between two vectors within the material is also necessary. This allows us to distinguish between pure rotation and actual deformation, ensuring that the material's stretch and distortion are accurately quantified.

To measure the change in angle between vectors due to deformation, we compute the scalar product between two vectors before and after deformation. Let $\boldsymbol{dX_1}$ and $\boldsymbol{dX_2}$ be two infinitesimal vectors starting at point $\boldsymbol{X}$, and $\boldsymbol{dx_1}$ and $\boldsymbol{dx_2}$ be their deformed states. The scalar product of the deformed vectors can be written as:

$$\boldsymbol{dx_1}.\boldsymbol{dx_2} = (\mathbb{F}\boldsymbol{dX_1})^{\boldsymbol{T}}.(\mathbb{F}\boldsymbol{dX_2}) = \boldsymbol{dX_1^T}\mathbb{F}^{\boldsymbol{T}}\mathbb{F}\boldsymbol{dX_2} = \boldsymbol{dX_1^T}\boldsymbol{\epsilon_c}\boldsymbol{dX_2} \tag{3.14}$$

In Equation 3.14, the term $\mathbb{F}^T\mathbb{F}$ is known as the right Cauchy-Green deformation tensor, denoted by $\boldsymbol{\epsilon_c}$. It encapsulates the stretch and rotation of the material during deformation. However, when the angle remains unchanged, $\boldsymbol{\epsilon_c}$ reduces to the identity matrix, indicating that the deformation is purely a rigid body motion (i.e., a combination of rotation and translation without stretching or distortion). To capture the actual deformation (excluding any rigid body motion), we compute the angle difference between the vectors in the rest and deformed configurations. This leads to the Green-Lagrange strain tensor $\boldsymbol{\epsilon_g}$ defined as:

$$\boldsymbol{\epsilon_g} = \frac{1}{2}(\boldsymbol{\epsilon_c} - \mathbb{I}) \tag{3.15}$$

The expression of the Green-Lagrange strain tensor in Equation 3.15 is obtained by measuring the difference between the two scalar products (in rest and deformed configurations), capturing the change in lengths and angles due to deformation, see Equation 3.16.

$$\boldsymbol{dx_1}.\boldsymbol{dx_2} - \boldsymbol{dX_1}.\boldsymbol{dX_2} = \boldsymbol{dx_1^T}\boldsymbol{\epsilon_c}\boldsymbol{dx_2} - \boldsymbol{dX_1}.\boldsymbol{dX_2} = \boldsymbol{dx_1^T}(\boldsymbol{\epsilon_c} - \mathbb{I})\boldsymbol{dx_2} = \boldsymbol{2dx_1^T}\boldsymbol{\epsilon_g}\boldsymbol{dx_2} \tag{3.16}$$

The Green-Lagrange strain tensor $\boldsymbol{\epsilon_g}$ quantifies the deformation by measuring how much the squared lengths of infinitesimal vectors and the angles between them deviate from their values in the rest configuration. It captures the linear and non-linear deformation components and satisfies $\boldsymbol{\epsilon_g} = 0$ when the angle is unchanged between both configurations. The linear and non-linear components of the strain $\boldsymbol{\epsilon_g}$ can be expressed in terms of the displacement field $\boldsymbol{\nabla u}$ using the definition of the deformation gradient, as follows:

$$\boldsymbol{\epsilon_g} = \frac{1}{2}(\mathbb{F}^{\boldsymbol{T}}\mathbb{F} - \mathbb{I}) = \frac{1}{2}((\boldsymbol{\nabla u} + \mathbb{I})^{\boldsymbol{T}}(\boldsymbol{\nabla u} + \mathbb{I}) - \mathbb{I}) = \frac{1}{2}(\boldsymbol{\nabla u^T \nabla u} + \boldsymbol{\nabla u^T} + \boldsymbol{\nabla u}) \tag{3.17}$$

This shows that the linear and non-linear components of the strain are $\boldsymbol{\epsilon_g^{linear}} = \frac{1}{2}(\boldsymbol{\nabla u^T} + \boldsymbol{\nabla u})$ and $\boldsymbol{\epsilon_g^{nonlinear}} = \frac{1}{2}(\boldsymbol{\nabla u^T \nabla u})$ respectively. When the displacement gradients are small, the non-linear component $\boldsymbol{\epsilon_g^{nonlinear}}$ can be ignored. As a result, the strain tensor

can be approximated following Equation 3.18, which is referred to as the infinitesimal strain tensor.

$$\epsilon_g \approx \epsilon_g^{linear} = \frac{1}{2}(\nabla u^T + \nabla u) \tag{3.18}$$

3.1.4 Constitutive law

In the previous sections, we defined the concepts of stress and strain. Stress quantifies the internal forces within a material per unit area, whereas strain measures the deformation or change in shape resulting from these forces. This section will delve into the relationship between stress and strain, known as the constitutive model. This model describes how stress correlates with strain based on the material's inherent properties, providing the material's response to applied forces. The simplest relationship between stress and strain is linear, often described by Hooke's Law. Hooke's Law states that, within the elastic limit of a material, the stress applied to a material is directly proportional to the resulting strain. This can be expressed mathematically as:

$$\sigma = K\epsilon \tag{3.19}$$

Where σ denotes the stress applied to the material, ϵ represents the resulting strain, and K is the stiffness matrix. It measures the material's stiffness or rigidity and indicates how much it will deform under a given stress. Hooke's Law assumes that the material deforms linearly, meaning that the deformation is directly proportional to the applied stress. The material will return to its original shape once the load is removed. This linear relationship holds if the material remains within its elastic limit and does not undergo permanent deformation. When dealing with isotropic materials under the assumption of small deformations, the constitutive model simplifies the stress-strain relationship. In this case, Hooke's Law applies, and the relationship between stress and strain is linear in terms of the strain tensor ϵ and the displacement gradient ∇u. This is because the linear approximation of the strain tensor in Equation 3.18 is used. For an isotropic material, the stress tensor σ is related to the strain tensor ϵ by:

$$\sigma = \lambda \mathrm{tr}(\epsilon)\mathbb{I} + 2\mu\epsilon \tag{3.20}$$

Where λ and μ are the Lamé parameters, $\mathrm{tr}(\epsilon)$ is the trace of the strain tensor, and $\mathbb{I}$ is the identity matrix. However, this linear relationship is insufficient when the displacements are not small. The material's behavior becomes nonlinear, and the linearized strain tensor no longer adequately describes the deformation. In such cases, hyperelastic models are

commonly used to deal with large deformations, where the stress-strain relationship is derived from a strain energy density function, denoted $\boldsymbol{W}$. The following briefly presents three examples of hyperelastic material models, each with its specific strain energy function.

3.1.4.1 Saint-Venant-Kirchhoff

The Saint Venant–Kirchhoff (SVK) material model is one of the simplest forms of hyperelastic models, serving as a direct extension of Hooke's law from linear elasticity to the realm of large deformations. It describes a homogeneous and isotropic material, meaning that its mechanical response is uniform across all points and in every direction. The material's resistance to deformation does not vary with location or orientation. In this model, the material properties are characterized by the Lamé parameters: λ and μ. These parameters are related to Young's modulus E and Poisson's ratio ν as follows:

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \quad \mu = \frac{E}{2(1+\nu)} \tag{3.21}$$

The strain energy density function W for the SVK material is expressed as:

$$\boldsymbol{W} = \frac{\lambda}{2}[\mathrm{tr}(\mathbb{E})]^2 + \mu \mathrm{tr}(\mathbb{E}^2) \tag{3.22}$$

Where $\mathbb{E}$ is the Green-Lagrange strain tensor, $\mathrm{tr}(\mathbb{E})$ denotes the trace of $\mathbb{E}$, and $\mathrm{tr}(\mathbb{E}^2)$ accounts for the shear deformation. By taking the first and second derivatives of the strain energy function $\boldsymbol{W}$ with respect to the Green-Lagrange strain tensor $\mathbb{E}$, one can derive the second-order Piola-Kirchhoff (PK2) stress tensor and the fourth-order material elasticity tensor, respectively.

3.1.4.2 Neo-Hookean

The Neo-Hookean model is a nonlinear extension of the SVK approach. Initially, the relationship between strain and stress in this model is linear, but as deformation increases, it deviates from linearity and the stress-strain curve eventually flattens out. This model overcomes the limitations of the SVK model, especially when dealing with large compressive strains. The strain energy density function for a Neo-Hookean material is expressed as:

$$\boldsymbol{W} = \frac{\mu}{2}(\mathrm{tr}(\mathbb{C}) - 3) - \mu \ln \boldsymbol{J} + \frac{\lambda}{2}(\ln \boldsymbol{J})^2 \tag{3.23}$$

Where $\mathbb{C}$ is the right Cauchy-Green deformation tensor defined in Equation 3.14, J is the determinant of the deformation gradient $\mathbb{F}$, λ and μ are the Lamé parameters expressed in Equations 3.21.

3.1.4.3 MooneyRivlin

Mooney-Rivlin is a hyperelastic model characterized by uncoupled deviatoric and volumetric behavior. It is designed to capture the complex, nonlinear elastic response of nearly incompressible materials under large deformations. The strain-energy density function $\boldsymbol{W}$ is expressed as:

$$\boldsymbol{W} = C_{01}(\boldsymbol{J}^{-\frac{2}{3}}\mathrm{I_C} - 3) + C_{10}(J^{-\frac{4}{3}}\mathrm{II_C} - 3) + \frac{\ln \boldsymbol{J}}{2D_1} \tag{3.24}$$

With C_{01}, C_{10} and D_1 being the material parameters, $\boldsymbol{J}$ is the jacobian matrix, while $\mathrm{I_C} = \mathrm{tr}(\mathbb{C})$ and $\mathrm{II_C} = \frac{1}{2}\left((\mathrm{tr}(\mathbb{C}))^2 - \mathrm{tr}\left(\mathbb{C}^2\right)\right)$ are the first and second invariants of the right Cauchy-Green deformation tensor $\mathbb{C}$, respectively. These invariants account for the deviatoric (shear) deformations, which are independent of changes in volume. The uncoupled approach of this model allows for separate consideration of volumetric and deviatoric behaviors, making it suited for modeling materials that experience significant shear deformations without corresponding changes in volume.

3.1.5 Problem formulation

In the earlier sections, we introduced the strain tensor, which links deformation to displacement, and the stress tensor, which represents the internal forces within a material. We also discussed the relationship between these two tensors with respect to the material properties. To achieve equilibrium between internal and external forces within a computational domain Ω, we use the following equation.

$$\mathrm{div}(\boldsymbol{\sigma}) + \boldsymbol{f_{ext}} = \rho \ddot{\boldsymbol{u}} \tag{3.25}$$

Where σ denotes the stress tensor, $\boldsymbol{f_{ext}}$ represents the external forces applied to the material, ρ is the density of the material, and $\ddot{\boldsymbol{u}}$ is the acceleration vector. This equation, known as the strong formulation of the equilibrium condition, expresses the fundamental principle of dynamics for every point in the body. We set the acceleration $\ddot{\boldsymbol{u}}$ to zero for the static case. This simplifies the equilibrium equation to:

$$\mathrm{div}(\boldsymbol{\sigma}) + \boldsymbol{f_{ext}} = 0 \tag{3.26}$$

Appropriate boundary conditions must be applied to ensure that this static equilibrium problem has a unique solution. Dirichlet boundary conditions are specified on a subset Γ_D of the boundary Γ of the domain Ω. These conditions dictate that:

$$\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{u_D}, \quad \forall \boldsymbol{x} \in \Gamma_D \tag{3.27}$$

Where u_D is a known displacement value. Neumann boundary conditions are also applied on another subset Γ_N of Γ. These conditions specify the surface forces as:

$$\boldsymbol{\sigma}(\boldsymbol{x}).\vec{n}(\boldsymbol{x}) = \boldsymbol{f_s}, \quad \forall \boldsymbol{x} \in \Gamma_N \tag{3.28}$$

With $\vec{n}(\boldsymbol{x})$ being the outward-pointing unit normal vector at the boundary point $\boldsymbol{x}$, and $\boldsymbol{f_s}$ represents the external surface forces. Dirichlet and Neumann boundary conditions ensure that the displacement and surface forces are properly accounted for, allowing for a well-posed static equilibrium problem.

Various numerical methods can be employed to solve equilibrium problems effectively. One crucial step in simplifying and solving these problems is transitioning from the strong formulation to the weak formulation. This transition involves reformulating the differential equation into a form that is more amenable to finite element methods and other approximation techniques. The weak form allows us to distinguish the forces applied at the domain's boundary (denoted $\boldsymbol{f_s}$) from the volume forces (such as the gravity), denoted $\boldsymbol{f_v}$. The process to obtain the weak form includes several key steps starting from the strong formulation of static equilibrium in Equation 3.26, we multiply both sides by a test function $\boldsymbol{v}$. The test function $\boldsymbol{v}$ belongs to a function space V and is chosen to satisfy specific boundary conditions appropriate to the problem. In finite element analysis, $\boldsymbol{v}$ is typically selected to be smooth and satisfy the problem's boundary conditions, such as being zero on Dirichlet boundaries. Then, we integrate both sides of the equation over the domain Ω:

$$-\int_\Omega \text{div}(\boldsymbol{\sigma}).\boldsymbol{v} d\Omega = \int_\Omega \boldsymbol{f_{ext}}.\boldsymbol{v} d\Omega \tag{3.29}$$

Next, by applying Green's formula and using the fact that $\boldsymbol{v}$ is chosen to be null on Γ_D, we simplify the Equation 3.29 to the following:

$$\int_\Omega \boldsymbol{\sigma} : \boldsymbol{\epsilon}(\boldsymbol{v}) d\Omega - \int_{\Gamma_N} \boldsymbol{f_s}.\boldsymbol{v} d\Gamma = \int_\Omega \boldsymbol{f_v}.\boldsymbol{v} d\Omega \tag{3.30}$$

Where $\sigma : \epsilon(v)$ denotes the double dot product between the stress tensor σ and the gradient of the test function v In this formulation, the test function v is required to satisfy the boundary conditions, and be appropriately chosen from the function space V. This weak

formulation is suitable for numerical solutions, such as the finite element method, which we will present in the next section.

3.2 The finite element method

In the previous section, we introduced the weak formulation of the equilibrium problem defined on a continuous domain. This weak form is well-suited for numerical analysis and provides a foundation for various computational techniques. Among these techniques, the finite element method (FEM) is one of the most widely used approaches for solving such problems. FEM discretizes the continuous domain into smaller, manageable elements and approximates the solution using piece-wise functions. This method allows for the efficient and accurate analysis of complex structures and materials. In this section, we will delve deeper into the finite element method, exploring its principles, formulation, and application to the weak formulation of equilibrium problems.

3.2.1 Domain discretization

In Section 3.1.5, we have reduced the order of the problem and explicitly formulated boundary terms to impose traction conditions on our system. See Equation 3.30. However, performing the necessary integrations over a continuous 2D or 3D field, particularly for complex and irregular shapes, can be challenging due to the lack of a precise numerical representation of the geometry. To overcome this, *discretization* is a crucial step of the FEM. It involves dividing the complex shape of an object into smaller, simpler geometric elements. As illustrated in Figure 3.2, these elements could be line segments in 1D, triangles and quadrilaterals in 2D, or tetrahedrons and hexahedrons in 3D. This simplifies the problem by breaking the global shape into trivial geometries, enabling easier numerical integration.

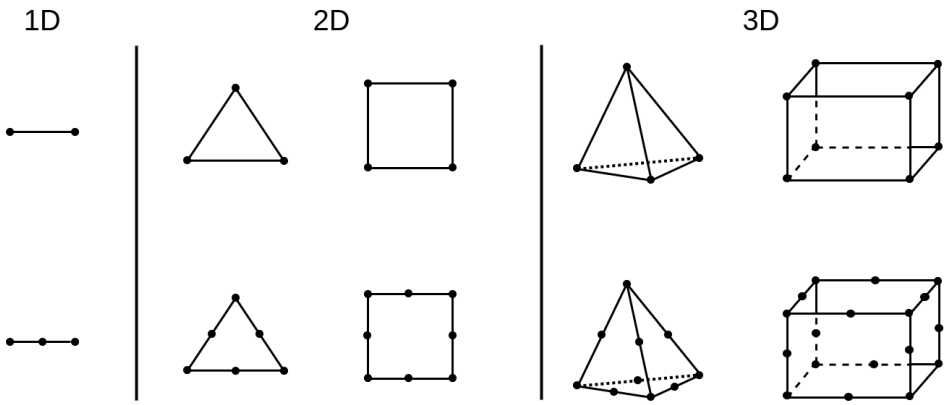


Figure 3.2: Example of linear and quadratic elements in 1D, 2D and 3D.

Additionally, the elements can vary in complexity. Linear elements are the simplest, with nodes only at the corners, leading to a linear approximation of the solution within each element, see Figure 3.2. In contrast, quadratic elements include additional nodes along the edges, allowing for a quadratic approximation and a more accurate representation of the geometry and solution. This distinction between linear and quadratic elements FEM impacts the accuracy and computational cost. The collection of these geometric elements, which results from the discretization process, is commonly referred to as a *mesh*. Inside each element of the mesh, shape functions are used to interpolate the value of a given point with respect to the element values. The continuous terms in the weak from (Equation 3.30) are substituted by a finite sum of piecewise continuous terms defined on each element. The displacement u and the test function v are replaced by their respective approximations u_e and v_e, which are restricted to within an individual element e. As a result, approximating the value of a field function at any position X within the initial domain Ω involves identifying the element e that contains X and then computing the approximation as:

$$\boldsymbol{u}_e^h(\boldsymbol{X}) = \sum_{i=0}^{n_e-1} \boldsymbol{N_i}(\boldsymbol{X})\boldsymbol{u_i} \tag{3.31}$$

With $\boldsymbol{u_i}$ the displacement value at the node i, n_e the number of nodes in the element e and $\boldsymbol{N_i}(\boldsymbol{x}) : \Omega \to \mathbb{R}$ is the i^{th} shape function of the element. Despite the existence of various types of shape functions, ranging from simple to highly complex definitions, for the sake of clarity and simplicity, we will illustrate linear (or trilinear) shape functions. For instance, first-degree polynomials. For linear triangular and tetrahedron elements, the nodal shape function $\boldsymbol{N_i}$ at node i is expressed as:

$$\boldsymbol{N_i}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) = \sum_{j+k+l\leq 1} \alpha_{jkl}\boldsymbol{x}^j\boldsymbol{y}^k\boldsymbol{z}^l \tag{3.32}$$

With α_{jkl} being real coefficients that satisfy $N_i = 1$ at node i and $N_i = 0$ on other nodes. In the case of linear quadrilateral and hexahedral elements, the shape function at node i is written as follows:

$$\boldsymbol{N_i}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) = \sum_{j,k,l\leq 1} \alpha_{jkl}\boldsymbol{x}^j\boldsymbol{y}^k\boldsymbol{z}^l \tag{3.33}$$

3.2.2 Building the system

In this section, we will detail the process of transitioning from the weak formulation to constructing a matrix system, which is essential for solving problems numerically. To

simplify the discussion, we will assume small displacements. This assumption allows us to approximate the stiffness matrix $\boldsymbol{K}$ explicitly, representing the relationship between displacements and internal forces. In many practical scenarios, however, the problem may involve material nonlinearity (a complex stress-strain relationship) or geometrical nonlinearity (a nonlinear strain-displacement relationship). In such cases, decoupling the material stiffness from the displacement field $\boldsymbol{u}$ becomes impossible. As a result, the problem must be solved iteratively, often using the Newton-Raphson algorithm, with the stiffness matrix $\boldsymbol{K}$ replaced by a tangent stiffness matrix updated at each iteration. When dealing with small displacements, the stiffness matrix $\boldsymbol{K}$ remains constant over time, allowing its inverse to be computed once at the beginning of the simulation, which significantly accelerates the computation. However, $\boldsymbol{K}$ must be recalculated at each simulation step for more complex materials or large deformations, capturing each point's dynamic influence on the mesh and the internal forces generated by their movements. Let $\boldsymbol{u_h}$ be the discretized version of the displacement $\boldsymbol{u}$, at a given point $\boldsymbol{X}$ it reads as:

$$\boldsymbol{u_h}(\boldsymbol{X}) = \sum_{i=1}^{N} \boldsymbol{N_i}(p).\boldsymbol{u_{hi}} \tag{3.34}$$

Let $\boldsymbol{f_h^s}$ and $\boldsymbol{f_h^v}$ be the discretized forms of the surface and volume, respectively. Using the shape function, these quantities can be expressed as:

$$\boldsymbol{f_h^s} = \sum_{i=1}^{N} \boldsymbol{N_i}(\boldsymbol{x}).\boldsymbol{f_{hi}^s} \qquad \boldsymbol{f_h^v} = \sum_{i=1}^{N} \boldsymbol{N_i}(\boldsymbol{x}).\boldsymbol{f_{hi}^v} \tag{3.35}$$

In Equations 3.34 and 3.35, $\boldsymbol{u_{hi}}$ is the displacement at the node i, $\boldsymbol{f_{hi}^s}$ and $\boldsymbol{f_{hi}^v}$ are the surface and volume forces at the node i, respectively. N the number of nodes in the *mesh*. Let $V_{h,0}$ the discretization of the Hilbert space $H_0^1(\Omega)$. Starting form the weak formulation in Equation 3.30, the problem is formulated as:

$$\exists ? \boldsymbol{u_h} \in V_{h,0} \text{such that} \int_\Omega \boldsymbol{\sigma}(\boldsymbol{u_h}) : \boldsymbol{\epsilon}(\boldsymbol{v_h}) d\Omega - \int_{\Gamma_N} \boldsymbol{f_h^s}.\boldsymbol{v_h} d\Gamma = \int_\Omega \boldsymbol{f_h^s}.\boldsymbol{v_h} d\Omega, \forall\, v_h \in V_{h,0} \tag{3.36}$$

Aiming to build a matrix system out of the problem formulated in Equation 3.36, The Galerkin theorem, which suggests replacing the test function with the shape function $N_n(X)$, is applied. This leads to:

$$\sum_{i=1}^{N} \boldsymbol{u_{hi}}(\int_{\Omega} \boldsymbol{\sigma}(N_i(X) : \boldsymbol{\sigma}(N_n(X))d\Omega) - \sum_{i=1}^{N} \boldsymbol{f_{hi}^{v}}(\int_{\Gamma_N} N_i(X).N_n(X)d\Gamma) = \sum_{i=1}^{N} \boldsymbol{f_{hi}^{v}}(\int_{\Omega} N_i(X).N_n(X)d\Omega) \tag{3.37}$$

In Equation 3.37, the domain is discretized, and the Galerkin theorem is applied to the problem formulation illustrated in Equation 3.36. This allows us to write the following system:

$$\boldsymbol{K}\boldsymbol{U_h} = \boldsymbol{F_h^s} + \boldsymbol{F_h^v} \tag{3.38}$$

With $\boldsymbol{K}$, $\boldsymbol{U_h}$, $\boldsymbol{F_h^s}$ and F_h^v being, respectively, the stiffness matrix, the displacement vector, the surface force vector, and the volume force vector. These quantities are expressed as:

$$\boldsymbol{K} = \begin{pmatrix} \int_{\Omega} \boldsymbol{\sigma}(\boldsymbol{N_1}(\boldsymbol{X})) : \boldsymbol{\epsilon}(\boldsymbol{N_1}(\boldsymbol{X}))d\Omega & \cdots & \int_{\Omega} \boldsymbol{\sigma}(\boldsymbol{N_1}(\boldsymbol{X})) : \boldsymbol{\epsilon}(\boldsymbol{N_N}(\boldsymbol{X}))d\Omega \\ \vdots & \vdots & \vdots \\ \int_{\Omega} \boldsymbol{\sigma}(\boldsymbol{N_N}(\boldsymbol{X})) : \boldsymbol{\epsilon}(\boldsymbol{N_1}(\boldsymbol{X}))d\Omega & \cdots & \int_{\Omega} \boldsymbol{\sigma}(\boldsymbol{N_N}(\boldsymbol{X})) : \boldsymbol{\epsilon}(\boldsymbol{N_N}(\boldsymbol{X}))d\Omega \end{pmatrix} \tag{3.39}$$

$$\boldsymbol{U_k} = \begin{pmatrix} \boldsymbol{u_{h1}} \\ \vdots \\ \boldsymbol{u_{hN}} \end{pmatrix} \qquad \boldsymbol{F_h^s} = \begin{pmatrix} \boldsymbol{f_{h1}^s} \\ \vdots \\ \boldsymbol{f_{hN}^s} \end{pmatrix} \qquad \boldsymbol{F_h^v} = \begin{pmatrix} \boldsymbol{f_{h1}^v} \\ \vdots \\ \boldsymbol{f_{hN}^v} \end{pmatrix} \tag{3.40}$$

3.2.3 Solving the system

In the previous section, we have built a linear system that links the forces to the displacement, as illustrated in Equation 3.38. Solving this system allows for determining the displacement field where internal forces exactly counterbalance the external forces acting on the material. For linear systems, where the stiffness matrix $\boldsymbol{K}$ is constant. This can be straightforwardly solved using any direct solver. However, in the case of nonlinear systems—where either the stress-strain relationship is complex or large deformations lead to significant changes in geometry—the problem becomes more intricate. To tackle these nonlinearities, the Newton-Raphson method is commonly employed. This iterative algorithm refines the displacement field through successive approximations until equilibrium is achieved. This process involves linearizing the system by breaking down the overall dis-

placement $\boldsymbol{u}$ into smaller, incremental displacements. At each iteration n, we aim to find a correction $\boldsymbol{\delta_u^n}$ that satisfies the linearized set of equations:

$$\dot{\boldsymbol{K}}^{n-1}(\boldsymbol{u_{n-1}})\boldsymbol{\delta_u^n} = \boldsymbol{r}(\boldsymbol{u_0} + \boldsymbol{\delta_u^{n-1}}) + \boldsymbol{f} \tag{3.41}$$

With $\dot{\boldsymbol{K}}^{n-1}$ represents the tangent stiffness matrix from the previous iteration, $\boldsymbol{r}(\boldsymbol{u_0} + \boldsymbol{\delta_u^{n-1}})$ is the internal elastic force based on the initial displacement $\boldsymbol{u_0}$ and the correction from the previous iteration, and $\boldsymbol{f}$ is the external force vector. The displacement is then updated as follows:

$$\boldsymbol{u_n} = \boldsymbol{u_{n-1}} + \boldsymbol{\delta_u^n} \tag{3.42}$$

This iterative process continues until the correction $\boldsymbol{\delta_u^n}$ becomes sufficiently small, ensuring that the system has converged to a solution that accurately reflects the equilibrium state of the problem.

3.3 Conclusion

In conclusion, this chapter has laid the groundwork for understanding the core principles of continuum mechanics and the FEM, essential for simulating the behavior of biological tissues. We began by introducing key concepts such as the deformation gradient, which characterizes how a material deforms at a local level, and the stress and strain tensors, which describe internal forces and the resulting deformations. The constitutive relations that link these quantities and define the material response were discussed, focusing on hyperelastic materials, which are critical for modeling soft tissues like the liver, where large deformations can occur. Additionally, we transitioned from the strong to the weak formulation of the governing equations, simplifying problem-solving. The weak formulation is suitable for numerical methods, particularly the FEM. The process of discretizing the domain, constructing shape functions, and assembling the system of equations was explained, along with the numerical methods required to solve these systems effectively. The fundamentals presented in this chapter are required for a comprehensive understanding of the thesis. These principles will be directly applied to build a biomechanical model of the liver where a hyperelastic material is employed for its nonlinear elastic properties.

CHAPTER 4

SOniCS

In this chapter, we introduce the first contribution of this thesis: SOniCS, a framework designed to simplify the implementation of hyperelastic models. SOniCS integrates FEniCS, a tool for solving partial differential equations, with SOFA, a framework for real-time simulation of deformable objects. This integration enables more efficient and user-friendly development of complex hyperelastic models, streamlining the simulation process for soft tissues and other deformable structures. The work presented here forms the foundation for developing a biomechanical liver model using advanced hyperelastic laws where the organ's geometry is segmented from the preoperative CT data. This model plays a crucial role in the thesis by enabling non-rigid registration between preoperative CT and intraoperative IVUS data. The work presented in this chapter led to the following publications:

- **S. El hadramy***, A. Mazier*, J. N .Brunet. J. S. Hale, S. Cotin, S. PA. Bordas. "SOniCS: Develop intuition on biomechanical systems through interactive error controlled simulations". *Engineering with Computers* 40, 1857–1876 (2024) [Mazier, Arnaud and El Hadramy, Sidaty et al. (2024)].
- **S. El hadramy***, A. Mazier*, J. N .Brunet. J. S. Hale, S. Cotin, S. PA. Bordas. "SOniCS: Interfacing SOFA and FEniCS for advanced constitutive models". FEniCS Conference, San Diego, 2022. [Mazier, Arnaud and El Hadramy, Sidaty et al. (2022)].

4.1 Introduction

Designing efficient finite element (FE) simulation software is a challenging task. Indeed, as FEM is a vast field, numerous pieces of software emerged to fill different gaps. For instance, commercial software such as Abaqus [Smith (2009)] or Ansys [DeSalvo et Swanson (1985)] focus on user-friendly GUIs (Graphical User Interface) guiding the user from pre-processing to post-processing. This has the advantage of allowing users to perform complex simulations with a relatively basic theoretical knowledge of FE. Meanwhile, other pieces of software focused on specific domains such as Gmsh [Geuzaine et Remacle (2009)] for FE meshing, Paraview [Ahrens et al. (2005)] for FE visualization, OpenFoam [Jasak et al. (2007)] for CFD (Computational Fluid Dynamic) simulations, OpenXFEM [Bordas et al. (2006)] for extended finite elements [Jansari et al. (2019)], collocation methods [Jacquemin et Bordas (2021)], meshfree methods [Nguyen et al. (2008)], multiscale problems [Talebi et al. (2013)], or material point methods (MPM) [Sinaie et al. (2017)]. This historical perspective explains the countless FE solvers which makes an exhaustive state-of-the-art review quasi-impossible. Before selecting one piece of FE software, user should consider the benefits and disadvantages associated with each (i.e., meshing, parallel support,

solvers, coding language, and visualization). However, generally, simplicity of use and the ability to easily test modeling hypotheses appear like the most important considerations in selecting such a computational tool.
In the medical simulation context, several specific aspects have to be considered.

- **The material model complexity**: Contrary to engineering materials such as steel or copper, the mechanical properties of living organs were only recently quantified [Gibbons (1934); Payan et Ohayon (2017)] and show immense variability [Mihai et al. (2017)]. Various models have been proposed, e.g., anisotropic [Elouneg et al. (2021); Zhou et Fung (1997)], hyperelastic [Martins et al. (2006); Zeraatpisheh et al. (2021)], viscoelastic [Ehlers et Markert (2001); Urcun et al. (2021b)], or poroelastic [Lavigne et al. (2022); Simon (1992)] to accurately depict their complex mechanical behaviors. Indeed, predicting the deformations of bio-materials can only be achieved through complex material models (sometimes even multi-scale), which are rarely implemented in commercial software. One major difficulty of this implementation remains the linearization of highly nonlinear equations. For instance, hyperelasticity equations can be written as a minimization of a tensorial function. In most cases, this minimization is solved using gradient-descent algorithms requiring the first and second derivatives of the functional. Therefore, obtaining such high-order nonlinear derivatives is not straightforward and can be prone to manual errors.

- **The complexity of the simulation**: In addition to the complexity of the material model, the simulation setup itself can be problematic. For instance, the material parameters are patient-specific and require data-driven or inference methods [Han et al. (2011); Urcun et al. (2021a)]. The simulations can also (partially) involve unknown boundary conditions [Tagliabue et al. (2021)], contact with other organs [Courtecuisse et al. (2014)], or surgical tools [Cotin et al. (1999)]. The problem can also require multi-physics models (such as FSI [Borazjani (2013)] (Fluid-Structure Interactions)) or incompressibility [Mazier et al. (2022)], where classical displacement-based finite elements are prone to locking.

- **The error control and uncertainty of the solution**: When dealing with biomechanical simulations, several uncertainties always arise from the material parameters, loads, geometry, or boundary conditions. This is mainly due to the difficulty in estimating the mechanical properties through ex-vivo methods or the topology of biological tissues using medical imaging. Similarly, error control and mesh adaptivity are necessary to ensure homogeneous convergence of the solution over the domain

and that the mesh is optimal given a quantity of interest [Allard et al. (2012)]. Therefore, quantifying the uncertainty or controlling the error on quantities of interests often requires making a very large number of simulations [Rappel et al. (2019)] which is incompatible with surgical timing [Hauseux et al. (2017)]. Meanwhile, those approaches could be functional in clinical settings by using accelerated simulations [Bui et al. (2018)] to build surrogate models or/and machine learning models for faster solutions of those highly non-linear parametric problems.

- **The real-time aspect**: In addition to the previous points, the numerical simulation's run time is crucial for clinical environments. When performing an operation, the surgeon cannot, in general, spend minutes waiting for the model predictions. Eventually, this aspect is also applicable to artificial intelligence. Indeed, in order to build an efficient machine-learning model, a significant amount of data is necessary. Using numerical simulations to create synthetic data is now standard and directly dependent on the simulation time [Brunet et al. (2019)]. Consequently, the run-time of the simulation can be considered a principal feature of biomechanics simulations, and indeed, of any non-intuitive non-linear problems subject to significant uncertainties in loading, boundary and initial conditions, and parameters.

- **The interaction with the user**: In a surgical simulation setting, external variables can impact the simulation during the execution. For example, the exact movement of the surgeon's tool influences the simulation at run-time. Thus, the parameters of the simulation must be tuned, "live", to integrate interactions between the user and the simulation [Niroomandi et al. (2013); Wu et al. (2014)].

- **The visual rendering**: Depending on the research field, visualization can play a critical role in understanding the results. For instance, in the computer graphics community, photo-realistic visualization is one of the main objectives [Gilles et al. (2011); Malgat et al. (2015)]. Contrastingly, in the mechanical engineering culture, visualization is a manner of extracting and understanding a solution or data set (i.e., stress or displacement fields) quantitatively. For medical simulations, an optimal solution must combine both the accuracy of the results and photo-realistic rendering citelecting the clinical ground truth all within clinical time frames [Guo et al. (2021)].

According to the state-of-the-art, SOFA [Faure et al. (2012)] (Simulation Open Framework Architecture) appears to be a suitable compromise. Indeed, SOFA employs efficient rendering while providing the possibility to interact in real-time with the running simulation. One can note that real-time computing is only possible depending on the complexity

of the problem. Indeed, using excessive numbers of degrees of freedom (DOFs) or solving a highly nonlinear problem cannot result in a real-time simulation (without using model order reduction [Chinesta et al. (2013); Goury et al. (2016)] or machine learning [Brunet et al. (2019))]. SOFA can also manage complex simulations through an efficient implementation of contact [Courtecuisse et al. (2010b); Duriez et al. (2004)], for example, or enabling multi-physics coupling. At the time of writing, only a few finite elements are available in SOFA. We remark that in this manuscript, the finite element depends on to the choice of basis functions via the Ciarlet triple (cell topology, polynomial space, dual basis) plus a specific choice of weak form. A similar issue is observed for material models where numerous implementations only focused on a 3-dimensional isotropic behavior. Therefore, coding a new finite element (Ciarler triple + weak form) in SOFA requires advanced C++ skills that may discourage individuals from using the software.

To alleviate the problem of complex material models, FEniCS [Alnaes et al. (2015)] seems like an appropriate solution. Indeed, FEniCS may not possess all of SOFA's features, but definitely overcomes SOFA's capabilities for material model complexity. With FEniCS, the user can generate any material model, regardless of the element's geometry or interpolation. Plus, it authorizes an export of the pertinent finite element tensors in C code to be efficiently plugged into SOFA. The benefits of the synergy are considerable. By using SOFA's interactivity and real-time features, the user can easily prototype a real-time simulation. Indeed, by modifying "live" various boundary conditions, geometries, or topologies, the user can effortlessly and rapidly verify modeling hypotheses for a specific problem. Combined with the specificities of FEniCS, the user can additionally smoothly prototype complex material models for modeling elaborate scenarios. Such feature has already been used by coupling FEniCS and Acegen but with different objectives [Lengiewicz et al. (2021)]. To the best of our knowledge, this work is the first to use FEniCS code generation capabilities for such an endeavor and is the first coupling between FEniCS and SOFA.

This chapter has the following outline. We will first briefly introduce SOFA and FEniCS, highlighting the relative advantages and design choices in each. Section 4.2 will detail the plugin functionalities and a short tutorial for importing a Saint Venant-Kirchhoff model from FEniCS in SOFA. Then, section 4.3 will focus on confirming our implementation for various numerical tests. We will use a manufactured solution in 4.3.1 as validation and compare our solutions for a cantilever beam problem with SOFA in 4.3.2. The last test consists of implementing a new material model (Mooney-Rivlin) in SOFA and benchmarking it with FEBio [Maas et al. (2012)] in section 4.3.3. Finally, in the last section 4.4, we will use our plugin in a complex haptic simulation that cannot be implemented in FEniCS, using a custom material model inexistent in SOFA.

4.1.1 SOFA

SOFA was created in 2007 by a joint effort from Inria, CNRS, USTL, UJF, and MGH. This piece of software aims to provide an efficient framework dedicated to research, prototyping, and the development of physics-based simulations. It is an open-source library distributed under the LGPL license, hosted on GitHub [1], and developed by an international community. SOFA is modular. Users can create public or private plugins to include additional features.

SOFA is a C++ library, including Python wrappers for a user-friendly prototyping interface. It was originally designed for deformable solid mechanics but has been extended to various domains such as robotics, registration, fluid simulations, model-order reduction, and haptic simulations [Courtecuisse et al. (2010a); Duriez (2013); Duriez et al. (2006)]. SOFA exhibits many attractive features, but among them, the combination of multi-model representations and mappings differentiate it from other software.

Multi-model representation: Most classical FE software uses an identical discretization for the whole model. Consequently, if one user wants to use the mesh along a contact surface, the FE mesh will undergo the same in the contact region. It can induce slow simulations for solving the FE system while the user was initially only interested in the contact part. Conversely, utilizing a multi-model representation approach, users can split the principal model into three distinct sub-models: deformation, collision, and visual. Thus, the user can decide to have high-fidelity deformations with flawed contact detection while maintaining a fine rendering, or vice-versa. Similarly, an object can be made of several deformation models. For example, one can model a muscle by the interaction of FE 3D tetrahedra for the volume and 1D beams for modeling ligaments, using 2 different solvers.

Mappings: In SOFA, the "mappings" are responsible for the communication between the different models. The models have parent-child relationships constructing a hierarchy (and a DOF hierarchy by extension). It enables propagating the positions, velocities, accelerations, and forces across the different models. For example, if the contact model calculates a force, it is mapped on the deformation model that will communicate back the computed displacement.

Finally, by combining the multi-model representation with the mappings, SOFA can build complex real-time simulations with high-fidelity rendering. In addition to a scene-graph structure and visitors (responsible for going through the model hierarchy) implementation, it can account for interactivity with the users.

[1] https://github.com/sofa-framework/sofa

Despite the advantages provided by SOFA, some drawbacks have to be acknowledged. First, in terms of solid mechanics simulations, only a few elements and material models are available. Indeed, SOFA only proposes Lagrange linear elements, and the cell topologies are limited to segments, triangles, quadrangles, tetrahedra, and hexahedra. The following material models are implemented: Boyce and Aruda [Arruda et Boyce (1993)], Costa [Costa et al. (2001)], isotropic and anisotropic Hookean, Mooney Rivlin (2 invariants) [Mooney (1940)], classical and stabilized Neo-Hookean, Ogden [Ogden (1972)], Saint Venant-Kirchhoff, and Veronda Westman [Veronda et Westmann (1970)]. Despite a reasonable number of mechanical models, a few of them are actually implemented for each cell topology. Secondly, the benefits provided by the mappings can also turn out to be a disadvantage when it comes to implementation. Indeed, the structure of the mappings is usually complex for unexperimented C++ users, and the mechanical tensors such as the Cauchy-Green or Piola-Kirchhoff are rarely computed. The two previous drawbacks are associated with the same flaw: the strong coupling between the material models and the topology of the element assumed within SOFA's architecture. This coupling implies that changing an element's topology or interpolation will involve a new mapping or the rewriting of the material model, even in the case of a similar material model.

4.1.2 The modular mechanics plugin (Caribou)

The initial goal of the plugin (called Caribou[2] at the time of writing) was to quickly implement new shape functions and their derivatives for different Immersed-Boundary and meshless domain discretization while keeping the compatibility with the existing SOFA surgical simulations [Brunet (2020)]. Besides, the plugin enabled to effortlessly implement different volumetric quadrature schemes and several hyperelastic material models. Hence, the software design had to be generic enough to combine all the previous 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. Hence, the plugin was made as an extension to SOFA, bringing a redesigned software architecture.

In the plugin, the authors implemented a compile-time polymorphism design using generic C++ template programming. The idea is to write the code as close as possible to equations found in traditional FE books. Then, the C++ compiler optimizes the set of operations executed during the simulation while keeping an object-oriented code. In this design, the "Element" concept was created as a generic computational class that would be inherited by all element types. Similarly to OpenXFEM++ [Bordas et al. (2006)], it pro-

[2] https://github.com/mimesis-inria/caribou

vides a flexible implementation to add interpolation and quadrature numerical procedures quickly. 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 to optimize the computation.

Finally, the plugin allows the creation of additional material models by simply defining three methods per material: the strain energy density function, the second Piola-Kirchhoff stress tensor function, and its derivative functions. These three functions are evaluated at a given integration point automatically provided by the plugin. This design delivers an undeniable advantage: writing a new material model is now independent of the topology and integration scheme. However, it comes with a non-negligible cost. The author of the new material model has to manually differentiate the strain energy twice and write it in C++. This manual intervention is error-prone and can quickly become a substantial drawback for complex materials.

4.1.3 FEniCS

The FEniCS Project (FEniCS) [Alnaes et al. (2015)] is a collection of tools for the automated solution of partial differential equations using the finite element method. Like SOFA, the FEniCS components are distributed under open-source licenses (LGPL v3 or later, and MIT) and development is hosted on GitHub[3].

A distinguishing feature of FEniCS is the ability to allow the user to write variational of weak formulations of finite element methods in a high-level Python-based domain-specific language (DSL), the Unified Form Language (UFL) [Alnæs et al. (2014)]. Subsequently, that high-level description can be compiled/transformed using the FEniCS Form Compiler (FFC) [Kirby et Logg (2006)] into low-level and high-performance kernels. These kernels can calculate the corresponding local finite element tensor for a given cell in the mesh. UFL is also used by other finite element solvers with independently developed automatic code generation capabilities, notably Firedrake [Rathgeber et al. (2016)], Dune [Bastian et al. (2021)]. The AceGen software also supports automatic differentiation and support for automaticaly generating low-level code [Korelc (2002)]. Compared with SOFA, FEniCS is limited in scope; its primary focus is the specification and solution of partial differential equations via the finite element method, leaving difficult problems like mesh generation, post-processing and visualization to leading third-party packages such as Gmsh [Geuzaine et Remacle (2009)] and Paraview [Ahrens et al. (2005)].

In the context of implementing finite element models of hyperelastic materials, this automatic approach has a number of advantages over the traditional route used by most

[3] https://github.com/fenics

finite element codes (including, to some extent, SOFA); differentiating analytical expressions for the residual (first derivative) and Jacobian (second derivative) of the energy functional for the hyperelastic model, picking a suitable finite element basis, and then hand-coding the corresponding finite element kernels in a low-level language (C, Fortran, C++) for performance. Specifically:

1. The symbolic residual and Jacobian can be derived automatically using the symbolic differentiation capabilities of UFL. By contrast taking these derivatives by hand can be tedious and error-prone.

2. The compilation of the UFL description of the problem by FFC into the associated low-level kernels is entirely automated. Again, this step is often time-consuming and difficult to perform manually.

3. Because of the high-level description of the problem it is possible to experiment quickly with different concrete finite element formulations (material models, basis functions, element topology, etc.) without manually modifying low-level kernel code.

The potential of this high-level approach for solid mechanics were recognized early on in the development of FEniCS, with two chapters in the FEniCS Book [Narayanan (2012); Ølgaard et Wells (2012)] promoting this direction. Since then, FEniCS has been used in a large number of publications on the topic of hyperelastic large-deformation elasticity e.g. [Baroli et al. (2012); Nguyen-Thanh et al. (2019); Patte et al. (2022)]. Recently the FEniCS Project has undergone a major redevelopment, resulting in the new FEniCSx components; DOLFINx (the finite element problem solving environment, replacing DOLFIN), FFCx (the FEniCSx Form Compiler, replacing FFC) and Basix [Scroggs et al. (2022)] (a finite element basis function tabulator, replacing FIAT [Kirby (2004)]). UFL is largely unchanged from the version used in the old FEniCS components and Firedrake.

In this work we do not use DOLFINx. DOLFINx contains the basic finite element data structures and algorithms (e.g. meshes, function spaces, assembly, interfacing with linear algebra data structures in e.g. PETSc [Balay et al. (2022)]) and therefore there is a significant overlap with the functionality already available in SOFA. Directly interfacing DOLFINx and SOFA at the Application Programming Interface (API) level would be a significant technical challenge due to the substantial differences in their internal data structures. Dedicated weak coupling libraries such as PreCICE [Rodenberg et al. (2021)] could be an interesting alternative to API coupling, but it is not a path that we explore in this work.

Instead, the approach taken by SOniCS is to only use UFL and FFCx (which in turn depends on Basix) to convert the high-level description of the finite element problem into low-level C code, which are then called using SOFA's existing C++ finite element data structures and algorithms. Compared with coupling DOLFINx and SOFA directly, our approach creates a relatively light compile-time coupling between FEniCS (specifically, the generated C code) and SOFA (a large complex C++ code with many dependencies). Consequently, no additional runtime dependencies required for SOFA. This methodology will be familiar to users of the DOLFINx C++ interface where C finite element kernels are generated in a first step using UFL and FFCx and are then integrated into the DOLFINx solver in a second step through a standard compile/include/link approach. Without going into excessive detail, three changes in the redeveloped FEniCSx components have made SOniCS significantly easier to realize:

1. FFCx (and also DOLFINx) now support code generations using finite elements defined on quadrilateral and hexahedral cells, the default in SOFA.

2. Basix and FFCx have full support for Serendipity finite elements of arbitrary polynomial order following the construction of Arnold and Awanou [Arnold et Awanou (2011)]. Serendipity elements are used in SOFA and there was a desire to continue supporting Serendipity basis function due to their lower number of degrees of freedom per cell and generally lower number of local computations compared with standard tensor-product Lagrange elements. We remark that despite the widespread use of Serendipity elements in many solvers, they can only obtain optimal order convergence on affinely-mapped meshes, see e.g. [Arbogast et al. (2022)] for more details.

3. FFCx outputs C99 compliant code according to the UFCx interface, which is specified as a C header file included with FFCx. This is in contrast with FFC, which outputs C++03 compliant code conforming to an interface specified with a C++ header file. This switch makes it significantly easier to call FFCx generated kernels from libraries with a C Foreign Function Interface (FFI) such as Python and Julia, or any language which can easily call functions with a C ABI (e.g. Fortran). Although SOFA is a C++ libraries and could certainly call C++ generated kernels, the C interface is simpler to use, consisting only of structs containing basic native data types and functions.

4.2 Implementation details

4.2.1 UDL: from Finite Element (FE) model to Python code

In this section, we present more in-depth the SOniCS (SOFA + FEniCS). The plugin is available on GitHub[4]. We first introduce the procedure for defining the material model, the element, and the quadrature rule or degree using the UFL (Python) syntax. For simplicity, we only focused on a Saint Venant-Kirchhoff model. But the method can be generalized to all element types following the pipeline shown in figure 4.1. Secondly, we explain the methodology for converting the UFL script into efficient C kernels. Finally, we show the interface between the SOniCS plugin and SOFA, stating the conceptual and coding differences. For simplicity and as the modular mechanics plugin's name (Caribou) might change, we use the name SOFA to denote the combination of SOFA and the modular mechanics plugin.

The first step is to define the FE model using the UFL syntax in a Python script. As an example, in listing 4.1, we describe the 3D simplest hyperelastic model and element: Saint Venant-Kirchhoff with linear Lagrange finite elements on tetrahedra. In the context of hyperelastic simulations, we will exclusively describe features that users could be interested in customizing.

```
# material.py
from ufl import (Coefficient, Constant, Identity,
                 TestFunction, TrialFunction, inner, ds,
                 VectorElement, derivative, dx, grad,
                 tetrahedron, tr, variable)

# Function spaces
cell = tetrahedron
d = cell.geometric_dimension()
element = VectorElement("Lagrange", cell, 1)

# Trial and test functions
du = TrialFunction(element)  # Incremental displacement
v = TestFunction(element)  # Test function

# Functions
```

[4] https://github.com/mimesis-inria/caribou/tree/FeniCS-features

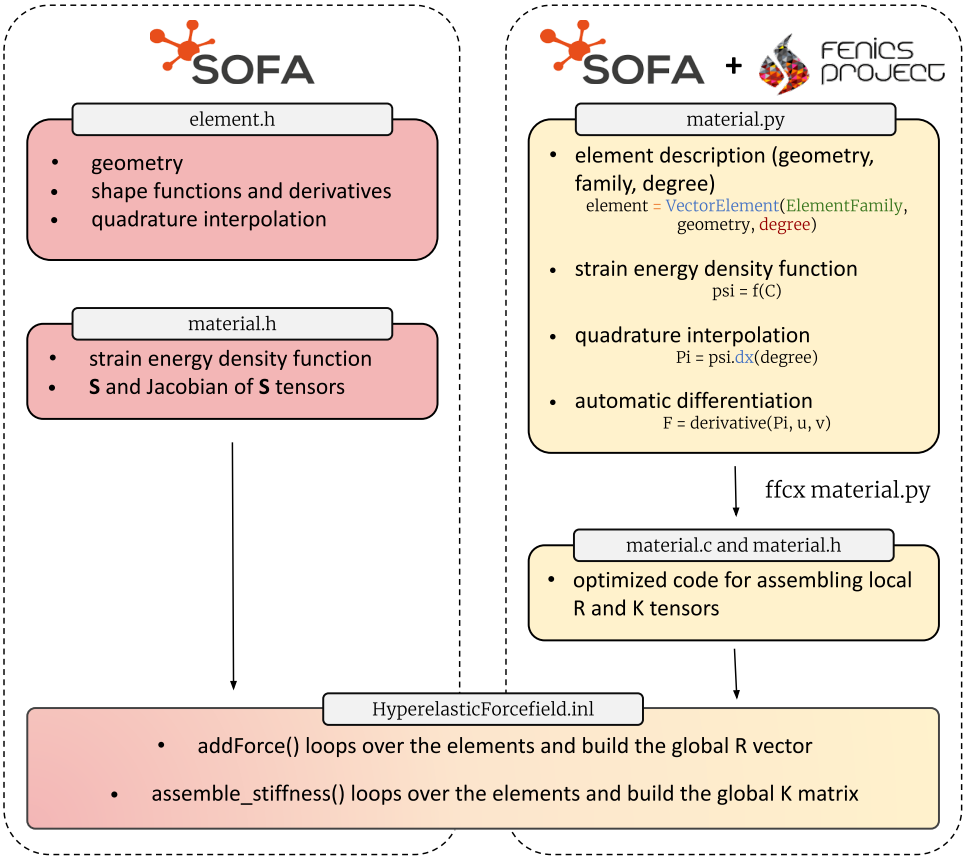


Figure 4.1: Description of the SOniCS pipeline (on the right) and differences with SOFA (on the left). In SOFA, each element has to be defined, embedding cell geometry, shape functions (including derivatives), and the quadrature scheme and degree. In SOniCS, it has been replaced by two Python lines of code for describing the element and its quadrature. The same benefit applies to the material model description. In SOFA, each material has to be created in a separate file stating its strain energy, derivating by hand the second Piola Kirchhoff tensor (S) and its Jacobian. It was replaced in SOniCS by only defining the strain energy of the desired material model in UFL. The derivative of the strain energy will then be automatically calculated using the FFCX module. Finally, both plugins share the same Forcefield methods for assembling the global residual vector ($\boldsymbol{R}$) and stiffness matrix ($\boldsymbol{K}$).

```
17 u = Coefficient(element)  # Displacement from previous
     iteration
18 B = Coefficient(element)   # Body forces
19 B = Coefficient(element)   # Traction forces
20
21 # Kinematics
```

```
22 I = Identity(d)  # Identity tensor
23 F = variable(I + grad(u))  # Deformation gradient
24 C = variable(F.T * F)  # Right Cauchy-Green tensor
25 E = variable(0.5 * (C - I))  # Green-Lagrange tensor
26
27 # Elasticity parameters
28 young = Constant(cell)
29 poisson = Constant(cell)
30 mu = young / (2 * (1 + poisson))
31 lmbda = young * poisson / ((1 + poisson) * (1 - 2 * poisson
       ))
32
33 # Stored strain energy density (compressible Neo-Hookean
       model)
34 psi = (lmbda / 2) * tr(E) ** 2 + mu * tr(E * E)
35
36 # Total potential energy
37 Pi = psi * dx(degree=1) - inner(B, u) * dx(degree=1) -
       inner(T, u) * ds(degree=1)
38
39 # First variation of Pi (directional derivative about u in
       the direction of v)
40 F = derivative(Pi, u, v)
41
42 # Compute Jacobian of F
43 J = derivative(F, u, du)
44
45 # Export forms
46 forms = [F, J, Pi]
```

Listing 4.1: Python code example (`material.py`) of a Saint Venant-Kirchhoff material model using Lagrange linear tetrahedron.

Element: After importing the necessary packages, we can define the element geometry. In listing 4.1, on line 10, we used a linear Lagrange tetrahedron. The user can easily modify different parameters of the element, such as the geometry, the family type, or the interpolation degree. For example, by only changing line 10 to

```
1 element = VectorElement("Serendipity", hexahedron, 2)
```

the element is now a quadratic Serendipity hexahedron. Note that `VectorElement` creates by default a function space of vector field equal to the spatial dimension. A complete list of the element available in the Basix documentation [Scroggs et al. (2022)].

Material model: By definition, boundary value problems for hyperelastic media can be expressed as minimization problems. For a domain $\Omega \subset R^3$, the goal is to find the displacement field $\boldsymbol{u} : \Omega \rightarrow R^3$ that minimizes the total potential energy Π. The potential energy is given by

$$\Pi(\boldsymbol{u}) = \int_\Omega \psi(\boldsymbol{u})\,\mathrm{d}x - \int_\Omega \boldsymbol{B}\cdot\boldsymbol{u}\,\mathrm{d}x - \int_{\partial\Omega} \boldsymbol{T}\cdot\boldsymbol{u}\,\mathrm{d}s, \tag{4.1}$$

where ψ is the elastic stored energy density, $\boldsymbol{B}$ is a body force (per unit reference volume), and $\boldsymbol{T}$ is a traction force (per unit reference area) prescribed on a boundary $\partial\Omega$ (of measure $\mathrm{d}s$) of the domain Ω (of measure $\mathrm{d}x$).

In listing 4.1, at line 37, we define the strain energy of a Saint Venant-Kirchhoff material model as:

$$\psi = \frac{\lambda}{2}\mathrm{tr}(\boldsymbol{E})^2 + \mu\mathrm{tr}(\boldsymbol{E}^2), \tag{4.2}$$

where λ and μ are Lamé material constants while $\boldsymbol{E}$ is the Green Lagrange strain tensor defined as $\frac{1}{2}(\boldsymbol{C}-\boldsymbol{I})$ where $\boldsymbol{I}$ is the identity matrix and $\boldsymbol{C}$ the right Cauchy-Green tensor $\boldsymbol{C} = \boldsymbol{F}^T\boldsymbol{F} = (\nabla\boldsymbol{u}+\boldsymbol{I})^T(\nabla\boldsymbol{u}+\boldsymbol{I})$. Therefore, we observe that tensor $\boldsymbol{E}$ is expressed with respect to the displacement $\boldsymbol{u}$, which is the unknown displacement field. Moreover, λ and μ are a function of the Young's modulus and of Poisson's ratio that are assumed to be known constants.

If the user would like to change the material model for Neo-Hookean with the strain energy density

$$\psi = \frac{\mu}{2}(I_C - 3) - \mu\ln(J) + \frac{\lambda}{2}\ln(J)^2. \tag{4.3}$$

It would only require in replacing line 31 with:

```
psi = (mu / 2) * (Ic - 3) - mu * ln(J) + (lmbda / 2) * (ln(
    J)) ** 2
```

in addition to previously defining the corresponding kinematics variables $J = \det(\boldsymbol{F})$ and $I_C = \mathrm{tr}(\boldsymbol{C})$.

```
J = det(F)
I_C = tr(C)
```

Quadrature rule: In listing 4.1, at line 37, when calculating the total potential energy it is also possible to choose the quadrature rule and the degree. In our example, we selected a quadrature degree of 1, triggering by default the Zienkiewicz and Taylor scheme [Zienkiewicz et al. (2014)] for tetrahedra. Hence, the user could also choose to

use a Gauss-Jacobi quadrature of degree 2 [Ralston et Rabinowitz (2001)] by replacing line 37 with:

```
Pi = psi * dx(degree=2, scheme="Gauss-Jacobi")  - inner(B, u) * dx(degree=2, scheme="Gauss-Jacobi") - inner(T, u) * ds(degree=2, scheme="Gauss-Jacobi")
```

4.2.2 FFCx: from Python code to efficient C kernels

In the particular case of static hyperelastic simulations, we solve the following non-linear system of equation

$$\boldsymbol{K}(\boldsymbol{u}) \cdot \mathrm{d}\boldsymbol{u} = \boldsymbol{R}(\boldsymbol{u}) - f(\boldsymbol{u}). \tag{4.4}$$

The $\boldsymbol{R}$ tensor is called the residual vector and is defined as the Gâteaux derivative of the total potential energy Π with respect to change in the displacement $\boldsymbol{u}$ in direction $\boldsymbol{v}$

$$\boldsymbol{R} = \frac{\mathrm{d}\Pi(\boldsymbol{u} + \epsilon \boldsymbol{v})}{\mathrm{d}\epsilon}|_{\epsilon=0}. \tag{4.5}$$

The tensor $\boldsymbol{K}$ is the Jacobian (also called stiffness in the context of mechanics) matrix and corresponds to the derivative of $\boldsymbol{R}$

$$\boldsymbol{K} = \frac{\mathrm{d}\boldsymbol{R}(\boldsymbol{u} + \epsilon \mathrm{d}\boldsymbol{u})}{\mathrm{d}\epsilon}|_{\epsilon=0}. \tag{4.6}$$

Solving the nonlinear system in equation 4.4 can be achieved using the Newton-Raphson algorithm that will iteratively solve a set of linear systems, assuming an initial guess $\boldsymbol{u}_n$

$$\boldsymbol{u}_{n+1} = \boldsymbol{u}_n - d\boldsymbol{u}. \tag{4.7}$$

The two tensors can be derived symbolically and exported using the UFL syntax with the function `derivative`, as shown at lines 40 and 43 in listing 4.1. A simple call to `ffcx` will create a .c and .h files containing the code for generating the local $\boldsymbol{R}$ and $\boldsymbol{K}$ tensors.

```
$ ffcx material.py
```

4.2.3 Integration in SOniCS

In SOFA, the definitions of the residual and stiffness tensors are carried out within a C++ file, `HyperelasticForcefield.cpp`. Each material model is in a separate file. So far, only Saint Venant-Kirchhoff and Neo-Hookean models have been implemented. The users can easily access those functionalities through Python wrappers:

```
node.addObject("SaintVenantKirchhoffMaterial",
    young_modulus=E, poisson_ratio=nu)
node.addObject('HyperelasticForcefield')
```

Listing 4.2: Python definition of a hyperealstic forcefield in SOFA using a Saint Venant-Kirchhoff material model.

The `HyperelasticForcefield` contains several functions, but we are particularly focusing on two of them. The `addForce` and `assemble_stiffness` functions are assembling the global residual and stiffness tensors respectively. Algorithm 1 details the `addForce` function, while algorithm 2 presents our reimplementation of the procedure. We did not detail the `assemble_stiffness` as it involves the exact same differences between the two implementations.

Algorithm 1: SOFA addForce function. The addForce function is in charge of assembling the global residual vector.

for *element in elements* **do**
- $X \leftarrow$ `element.positions` ▷ return the current positions of the element
- $R_{\text{global}} \leftarrow 0$ ▷ zero the global residual vector of dimension (DOFs ×3)
- $R_{\text{local}} \leftarrow 0$ ▷ zero the local residual vector of dimension (element DOFs ×3)
- **for** *quadrature in quadratures* **do**
  - $\det J \leftarrow \det($`quadrature.nodes`$)$ ▷ return the Jacobian of the quadrature nodes
  - $\mathrm{d}N \leftarrow$ `quadrature.nodes.shape_functions_derivatives` ▷ return the derivatives of the shape functions of the quadrature nodes
  - $w \leftarrow$ `quadrature.nodes.weights` ▷ return the weights of the quadrature nodes
  - $F \leftarrow X^T \cdot \mathrm{d}N$
  - $J \leftarrow \det(F)$
  - $C \leftarrow F^T \cdot F$
  - $S \leftarrow \mathrm{f}(C,$`MaterialParameters`$)$ ▷ return the second Piola-Kirchhoff depending on the material parameters and kinematics tensors
  - **for** *i in range(0, NumberOfNodesPerElement)* **do**
    - $\mathrm{d}x \leftarrow \mathrm{d}N[\mathrm{i}]^T$
    - $R_{local}[i] \leftarrow (\det J \cdot w) \cdot F \cdot S \cdot \mathrm{d}x$ ▷ allocate the result in the local residual vector
  - **end**
- **end**
- **for** *i in range(0, NumberOfNodesPerElement)* **do**
  - $R_{\texttt{global}}[\texttt{global}(i)] \leftarrow R_{\text{global}}[\texttt{global}(i)] - R_{\text{local}}[i]$ ▷ i indicates the element node index while `global`(i) denotes the global node index
- **end**

end

The structure is similar but we can still observe a few differences.

Algorithm 2: SOniCS addForce function. The addForce function is in charge of assembling the global residual vector.

1 **for** *element in elements* **do**
- $X \leftarrow$ `element.positions` ▷ return the current positions of the element
- $X_0 \leftarrow$ `element.rest_positions` ▷ return the initial positions of the element
- $B \leftarrow$ `gravity` ▷ return the body forces
- $T \leftarrow$ `element.forces` ▷ return the traction forces applied on the element
- $u \leftarrow X - X_0$
- $R_{\texttt{global}} \leftarrow 0$ ▷ zero the global residual vector of dimension (DOFs ×3)
- $R_{\texttt{local}} \leftarrow 0$ ▷ zero the local boundary conditions residual vector of dimension (element DOFs ×3)
- $R^{\texttt{bc}}_{\texttt{local}} \leftarrow 0$ ▷ zero the local residual vector of dimension (element DOFs ×3)
- `constants` $\leftarrow$ `MaterialParameters` ▷ return the material parameters
- $R_{\texttt{local}} \leftarrow \texttt{tabulate\_tensor}(R_{\texttt{local}}, u, B, \texttt{constants}, X_0)$
- $R^{\texttt{bc}}_{\texttt{local}} \leftarrow \texttt{tabulate\_tensor}(R_{\texttt{local}}, u, T, \texttt{constants}, X_0)$ **for** *i in range(0, NumberOfNodesPerElement)* **do**
  - $R_{\texttt{global}}[\texttt{global}(i)] \leftarrow R_{\texttt{global}}[\texttt{global}(i)] - (R_{\texttt{local}}[i] + R^{\texttt{bc}}_{\texttt{local}}[i])$ ▷ i indicates the element node index while $\texttt{global}(i)$ denotes the global node index
- **end**

2 **end**

- The new implementation of algorithm 2 is more concise and involves less visible tensorial operations because all those operations are efficiently hard coded in the C file provided by FFCx. For example, in algorithm 1, lines 5 to 17 were replaced in the new algorithm 2 by solely line 11.

- Algorithm 2 needs to have access to the initial position of the object and to the displacement vector. This was indeed not needed in the previous implementation since the modular mechanics plugin takes advantage of writing the deformation gradient only based on the current nodal coordinates $\boldsymbol{x}$:

$$\boldsymbol{F} = \boldsymbol{I} + \nabla_{\Omega_0}\boldsymbol{u} = \boldsymbol{I} + \nabla_{\Omega_0}(\boldsymbol{x} - \boldsymbol{x_0}) = \nabla_{\Omega_0}\boldsymbol{x}. \tag{4.8}$$

  ∇_{Ω_0} and $\boldsymbol{x_0}$ respectively denote the gradient and the nodal coordinates in the initial configuration, thus saving one extra vector operation.

- In the SOFA implementation, the boundary conditions and body forces are treated in separate files. In the new implementation of the Forcefield 1, the boundary conditions and body forces are now directly carried out in the Forcefield on lines 11

and 12. It avoids calling another function to loop again through every element of the object, thus speeding up the assembly of the residual vector.

Based on this new implementation, we created a new forcefield `HyperelasticForcefield_FEniCS` as close as possible to the existing syntax of `HyperelasticForcefield`. We also needed to tune the existing material definition to replace unnecessary calculations and allow us to read the corresponding .c file.

```
node.addObject("FEniCS_Material", material="
    SaintVenantKirchhoff", young_modulus=E, poisson_ratio=nu
    )
node.addObject('HyperelasticForcefield_FEniCS')
```

Listing 4.3: Python definition of a hyperelastic forcefield in SOniCS using a Saint Venant-Kirchhoff material model.

Finally, the last hurdle was the element definitions. Indeed, SOFA and FEniCS do not use the same vertices, edges, and facets ordering (as shown in figure 4.2). To avoid any conflict with the existing users of SOFA and solve the ordering issue, we proposed rearranging the topology indices, edges, and vertices and creating new elements. It ensured an accurate integration over the elements (especially for quadratic Serendipity integrating over the edges) and preserved an appropriate visualization. Those elements have been interfaced with the existing topology named `CaribouTopology`.

```
node.addObject('CaribouTopology', name='topology', template
    ="Hexahedron", indices=mesh.cells_dict['hexahedron'])

node.addObject('CaribouTopology', name='topology', template
    ="Hexahedron_FEniCS", indices=mesh.cells_dict['
    hexahedron'][:, [4, 5, 0, 1, 7, 6, 3, 2]])
```

Listing 4.4: Python definition of hexahedron topology in SOFA and SOniCS.

4.3 Numerical Examples

In this section we describe three numerical examples used for the validation of our SOniCS implementation. Every simulation described in this section can be reproduced and are available on GitHub[5]. We use the same domain description for each example while varying the boundary conditions and material parameters of each simulation.

[5] https://github.com/Ziemnono/SOniCS_validation

(a) FFCx/Basix. **(b)** SOFA.

Figure 4.2: Local numbering of element vertices and edges in both FEniCS and SOFA

Let Ω be a domain represented by a squared-section beam of dimensions $80 \times 15 \times 15\text{m}^3$, considered fixed on the right side ($\boldsymbol{u} = 0$ on Γ_D) while Neumann boundary conditions are applied on the left side (Γ_N), as shown in figure 4.3a and 4.3b.

Ω was discretized using two different geometrical elements using linear and quadratic interpolations. P1 and P2 elements stand for linear or quadratic tetrahedra, while Q1 and Q2 denote linear and quadratic hexahedra.

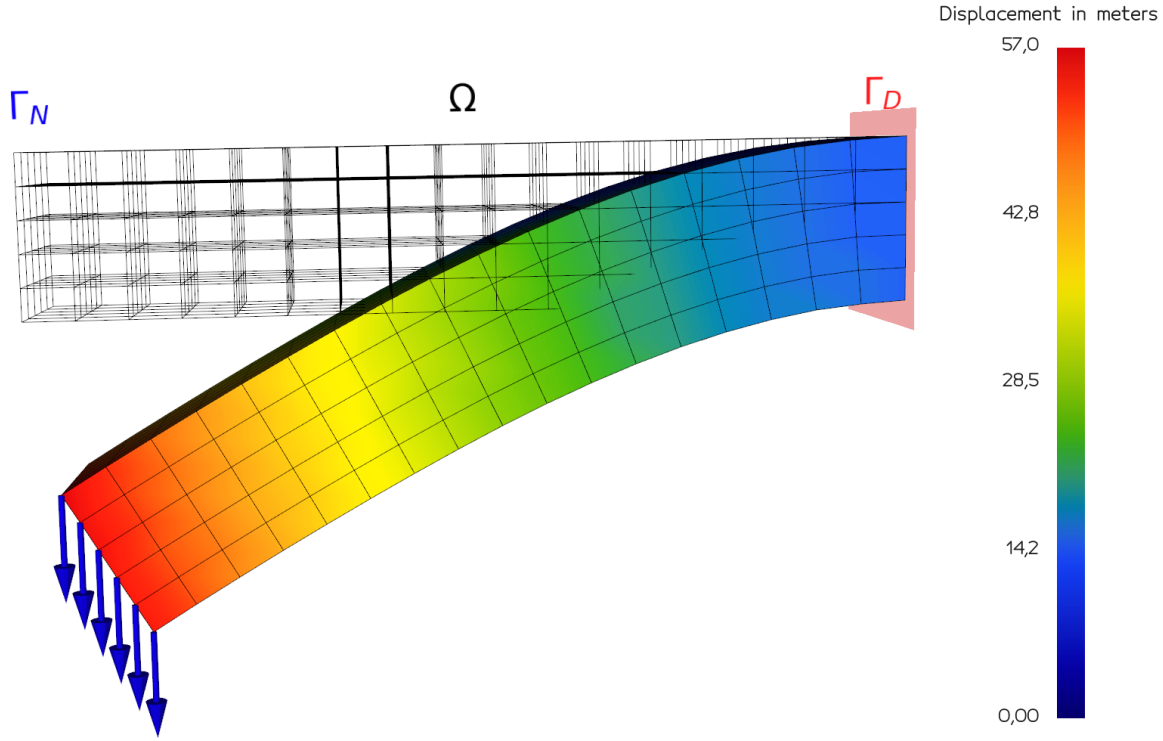
To solve each hyperelastic formulation described in equation 4.4, we used an identical implementation of the classical Newton Raphson (NR) solver for SOniCS, SOFA, and FEBio. The solver had the following parameters: a maximum of 25 iterations with a residual and displacement tolerance of 10^{-10}. In order to compare the running time of the two implementations, we, therefore, introduced the mean NR iteration time. We defined the NR iteration time as the duration for assembling and factorizing the system matrix, solving and propagating the unknown increment, updating, and computing the force and displacement residual. After checking that the same number of iterations have been achieved, we averaged the total time over the number of iterations needed for the solver convergence. All calculations were performed using an Intel® Core™ i5-6300HQ CPU @ 2.30GHz × 4 processor with a 16GiB memory and a NV117 / Mesa Intel® HD Graphics 530 (SKL GT2) graphics card.

We evaluated the soundness of the SOniCS solution using SOFA, FEBio, or a manufactured solution as the reference solution and computed the Euclidean relative L^2 error for the displacement and strain fields.

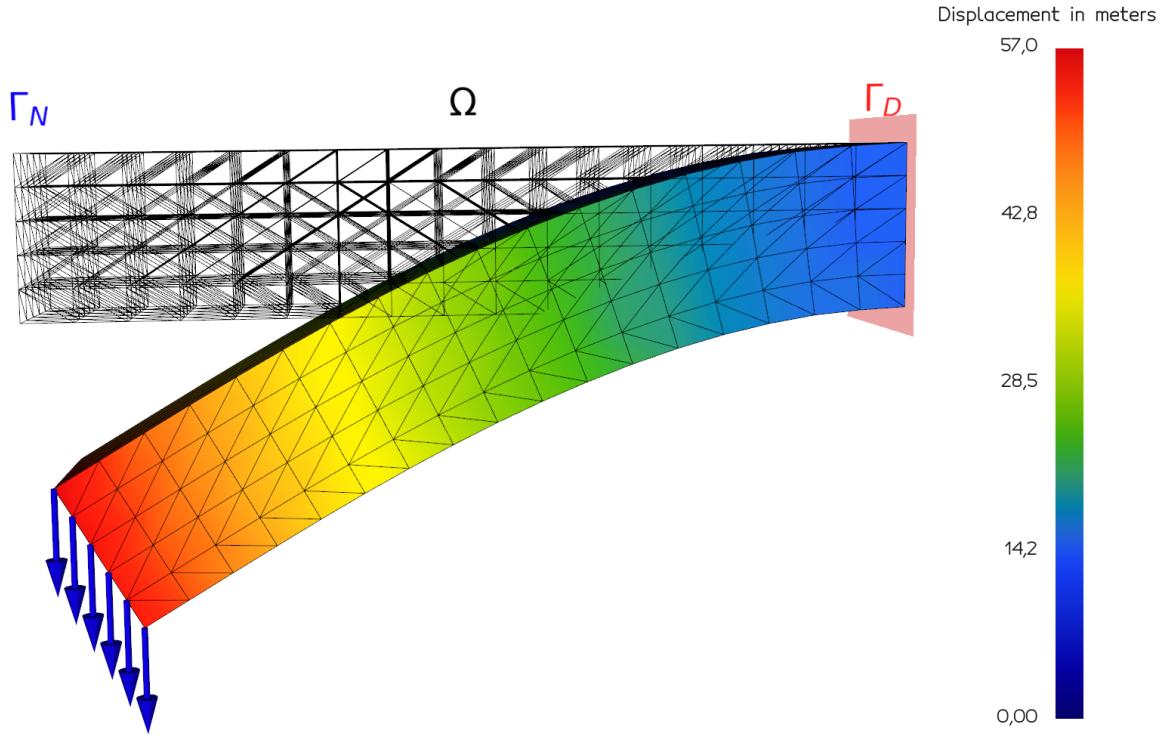
$$L^2_u(\boldsymbol{u},\boldsymbol{v}) = \frac{||\boldsymbol{u}-\boldsymbol{v}||_2}{||\boldsymbol{v}||_2}, \tag{4.9}$$

$$L^2_E(\boldsymbol{u},\boldsymbol{v}) = \frac{||\boldsymbol{E_u}-\boldsymbol{E_v}||_2}{||\boldsymbol{E_v}||_2}, \tag{4.10}$$

where $\boldsymbol{u}$ and $\boldsymbol{v}$ are the calculated displacement vectors for SOniCS and the reference implementation, respectively, $||\bullet||$ the Euclidean norm, and $\boldsymbol{E_u}$ the Green-Lagrange strain



(a) Hexahedral elements.



(b) Tetrahedral elements.

Figure 4.3: Cantilever beam domain discretization and displacement field. Ω is a domain represented by a squared-section beam of dimensions $80 \times 15 \times 15\text{m}^3$, considered fixed on the right side ($\boldsymbol{u} = 0$ on Γ_D) while Neumann boundary conditions are applied on the left side (Γ_N).

tensor of the displacement $\boldsymbol{u}$.

In this section, we first compare our solution with an analytical one: the manufactured solution. Then, we consider a clamped cantilever beam subject to Neumann boundary conditions and compare its deformation with the SOFA solution. Finally, using the same cantilever beam, we implemented a Mooney Rivlin model (not implemented in SOFA) using SOniCS and compared the solution with FEBio.

4.3.1 Manufactured solution

Aiming at code verification, the method of the manufactured solution consists in choosing an exact solution to the problem as an analytical expression [Chamberland et al. (2010)]. The chosen analytical expression is then inserted into the Partial Differential Equation (PDE) to determine the conditions that lead to this solution. In general, the manufactured chosen solution is expressed in simple primitive functions like $\sin(), \exp(), \tanh()$. In the context of hyperelastic equations, we considered the following manufactured solution for the displacement

$$\boldsymbol{u}(x,y,z) = \begin{bmatrix} 10^{-2} \cdot z \cdot e^{x} \\ 10^{-2} \cdot z \cdot e^{y} \\ 10^{-2} \cdot z \cdot e^{z} \end{bmatrix} \text{ on } \Omega. \tag{4.11}$$

Starting from the above-chosen displacement and using continuum mechanics laws, the relative analytical forces are applied as Neumann boundary conditions and deduced as follows.

$$\boldsymbol{F} = \boldsymbol{I_d} + \text{grad}(\boldsymbol{u}), \tag{4.12}$$

$$\boldsymbol{P} = \frac{\partial W}{\partial \boldsymbol{F}}, \tag{4.13}$$

$$\boldsymbol{f} = -\nabla \cdot \boldsymbol{P} \text{ on } \Gamma_N. \tag{4.14}$$

Where $\boldsymbol{F}$ is the deformation gradient, $\boldsymbol{I_d}$ is the identity matrix of dimension d, $\boldsymbol{P}$ is the first Piola-Kirchhoff stress tensor and $\boldsymbol{W}$ is the strain energy density depending on the material model constitutive law. We used a Saint Venant-Kirchhoff material (4.2) with a Young's Modulus of 3 kPa and a Poisson's ratio of 0.3, the computation of this solution was performed using Python Sympy package [Meurer et al. (2017)].

In this experiment, we generated 7 and 6 discretizations of P1 and P2 elements, respectively, with a decreasing element size in both scenarios. For each discretization, we

applied the relative analytical forces deduced from the manufactured solution (4.11) and used SOniCS Saint Venant-Kirchhoff material implementation with the same parameters as Sympy's to fill the domain. The displacements obtained were compared to the chosen analytical solution in equation (4.11) for each discretization. The results are presented in figure 4.4, and the error metrics are the relative errors presented in equations 4.9 and 4.10.

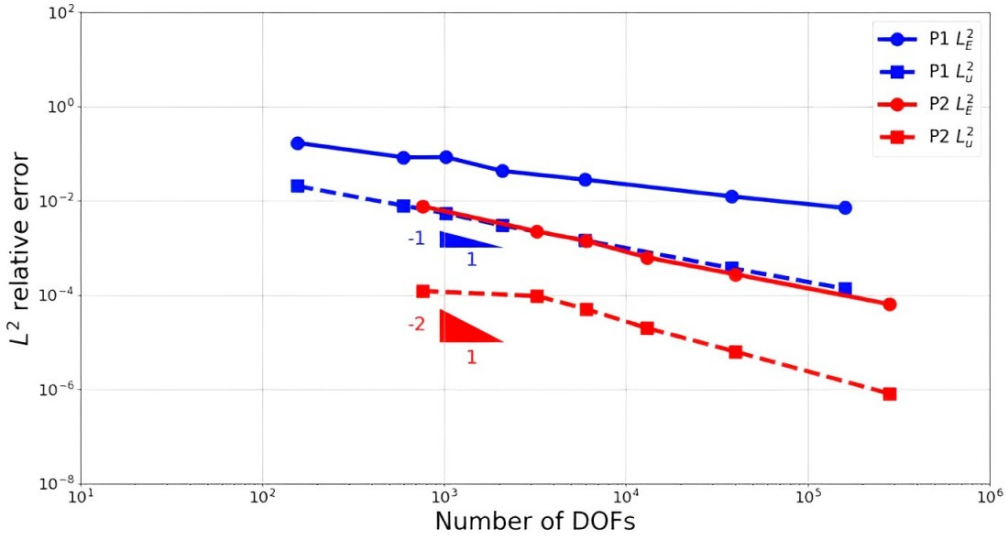


Figure 4.4: Plot of the mesh convergence analysis of the manufactured solution. The L^2 errors (L_u^2 for displacement and L_E^2 for strain) between the analytical and the SOniCS simulation is calculated for different number of DOFs with fixed parameters (E=3 kPa and ν=0.3) for P1 linear tetrahedra (blue) and P2 quadratic tetrahedra (red) elements.

4.3.2 Benchmark with SOFA

The cantilever beam deflection is a classical mechanical test case, as you can smoothly refine the mesh due to the simplicity of the geometry or modify its boundary conditions to fit real-life experiments. In this context, the beam was still clamped on the right side (natural Dirichlet condition on Γ_D) while Neumann boundary conditions were applied on the left side (Γ_N). To compare our SOniCS implementation, we model the deformation of the beam with two hyperelastic material models: Saint Venant-Kirchhoff and Neo-Hookean. We fixed the mechanical parameters and the Neumann boundary conditions equal to -10 Pa in the y direction until reaching sufficient large deformations with the same parameters as before: $E = 3$ kPa and $\nu = 0.3$.

This study aims at comparing the finite element solutions provided by SOniCS and SOFA under the same constraints in terms of computational and running time performances. To do so, we computed the relative L^2 error for the displacement and strain

fields ($L^2_u(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{SOFA}})$ and $L^2_E(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{SOFA}})$)between the SOniCS solution using SOFA as the reference solution.

The results obtained are presented in tables 4.1 and 4.2 for Saint Venant-Kirchhoff and Neo-Hookean materials, respectively.

Saint Venant-Kirchhoff material model					
Element	Number of DOFs	SOniCS mean NR iteration time (s)	SOFA mean NR iteration time (s)	$L^2_u(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{SOFA}})$	$L^2_E(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{SOFA}})$
P1	10935	0.387	0.438	4.10e-14	2.32e-16
P2	12705	0.810	0.808	3.49e-14	8.18e-15
Q1	10935	0.449	0.464	6.19e-13	3.76e-16
Q2	11772	0.839	0.942	3.81e-13	23.68e-14

Table 4.1: Relative error for displacement ($L^2_u(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{SOFA}})$) and strain ($L^2_E(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{SOFA}})$) defined in equations 4.9 and 4.10 and mean NR (Newton-Raphson) iteration time between SOniCS and SOFA for different element geometries and interpolation schemes using Saint Venant-Kirchhoff material model. P1 and P2 elements stand for linear or quadratic tetrahedra, while Q1 and Q2 denote linear and quadratic hexahedra.

Neo-Hookean material model					
Element	Number of DOFs	SOniCS mean NR iteration time (s)	SOFA mean NR iteration time (s)	$L^2_u(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{SOFA}})$	$L^2_E(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{SOFA}})$
P1	10935	0.391	0.428	2.17e-14	7.78e-14
P2	12705	0.826	0.852	1.40e-14	2.18e-20
Q1	10935	0.471	0.478	4.92e-13	5.77e-16
Q2	11772	0.826	0.864	1.64e-13	2.17e-14

Table 4.2: Relative error for displacement ($L^2_u(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{SOFA}})$) and strain ($L^2_E(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{SOFA}})$) defined in equations 4.9 and 4.10 and mean NR (Newton-Raphson) iteration time between SOniCS and SOFA for different element geometries and interpolation schemes using Neo-Hookean material model. P1 and P2 elements stand for linear or quadratic tetrahedra, while Q1 and Q2 denote linear and quadratic hexahedra.

4.3.3 Benchmark with FEBio

FEBio is an open-source finite element package specifically designed for biomechanical applications. It offers modeling scenarios, a wide range of constitutive material models, and boundary conditions relevant to numerous research areas in biomechanics. In this section, FEBio was used to compute the same scenarios as in section 4.3.2 to evaluate the trustworthiness of SOniCS. A more advanced constitutive material model, Mooney Rivlin, was introduced for this purpose

$$\psi = C_{01}\left(\overline{\mathrm{I}_C} - 3\right) + C_{10}\left(\overline{\mathrm{II}_C} - 3\right) + \frac{K}{2}\ln(J). \tag{4.15}$$

Where C_{01}, C_{10}, and K are the material constants in addition to the modified invariants $\overline{\mathrm{I}_C} = J^{-\frac{2}{3}}\,\mathrm{I}_C$, $\overline{\mathrm{II}_C} = J^{-\frac{4}{3}}\,\mathrm{II}_C$ defined based on the classic invariants $\mathrm{I_C} = \mathrm{tr}(\mathbf{C})$, $\mathrm{II_C} = \frac{1}{2}\left((\mathrm{tr}(\mathbf{C}))^2 - \mathrm{tr}\left(\mathbf{C}^2\right)\right)$. In order to obtain sufficiently large deformations, we chose the following material parameters: $C_{01} = 2000\,\mathrm{Pa}$, $C_{10} = 100\,\mathrm{Pa}$, and $K = 1000\,\mathrm{Pa}$.

Tables 4.3, 4.4, 4.5 show the results obtained for Saint Venant-Kirchhoff, Neo-Hookean and Mooney Rivlin material models and considering the four discretizations implemented so far in SOniCS. The error evaluation is still based on the mean relative error defined in equation 4.9 using FEBio as the reference while using a Newton Raphson solver with the same characteristics in both cases.

Saint Venant-Kirchhoff material model					
Element	Number of DOFs	SOniCS mean NR iteration time (s)	FEBio mean NR iteration time (s)	$L^2_u(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{FEBio}})$	$L^2_E(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{FEBio}})$
P1	10935	0.387	0.401	6.01e-10	4.23e-7
P2	12705	0.810	0.991	0.08	0.117
Q1	10935	0.449	0.508	4.19e-10	2.96e-6
Q2	11772	0.839	0.102	0.13	0.189

Table 4.3: Relative error for displacement ($L^2_u(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{FEBio}})$) and strain ($L^2_E(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{FEBio}})$) defined in equations 4.9 and 4.10 and mean NR (Newton-Raphson) iteration time between SOniCS and FEBio for different element geometries and interpolation schemes using Saint Venant-Kirchhoff material model. P1 and P2 elements stand for linear or quadratic tetrahedra, while Q1 and Q2 denote linear and quadratic hexahedra.

Neo-Hookean material model					
Element	Number of DOFs	SOniCS mean NR iteration time (s)	FEBio mean NR iteration time (s)	$L_u^2(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{FEBio}})$	$L_E^2(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{FEBio}})$
P1	10935	0.391	0.458	6.53e-10	7.33e-7
P2	12705	0.826	0.101	1.5e-2	3.16e-2
Q1	10935	0.471	0.523	8.08e-10	1.02e-6
Q2	11772	0.826	0.908	0.09	0.13

Table 4.4: Relative error for displacement ($L_u^2(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{FEBio}})$) and strain ($L_E^2(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{FEBio}})$) defined in equations 4.9 and 4.10 and mean NR (Newton-Raphson) iteration time between SOniCS and FEBio for different element geometries and interpolation schemes using Neo-Hookean material model. P1 and P2 elements stand for linear or quadratic tetrahedra, while Q1 and Q2 denote linear and quadratic hexahedra.

Mooney Rivlin material model					
Element	Number of DOFs	SOniCS mean NR iteration time (s)	FEBio mean NR iteration time (s)	$L_u^2(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{FEBio}})$	$L_E^2(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{FEBio}})$
P1	10935	0.662	0.804	2.49e-9	6.81-8
P2	12705	0.102	0.112	9.97e-3	8.28e-2
Q1	10935	0.818	0.910	10.92	14.71
Q2	11772	0.101	0.125	4.23e-2	1.229e-1

Table 4.5: Relative error for displacement ($L_u^2(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{FEBio}})$) and strain ($L_E^2(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{FEBio}})$) defined in equations 4.9 and 4.10 and mean NR (Newton-Raphson) iteration time between SOniCS and FEBio for different element geometries and interpolation schemes using Mooney Rivlin material model. P1 and P2 elements stand for linear or quadratic tetrahedra, while Q1 and Q2 denote linear and quadratic hexahedra. The large errors obtained for Q1 elements are further discussed in section 4.5.

4.4 Haptic simulation

In the context of numerical surgical simulations, robot haptic feedback has been shown to be a consistent tool for drastically improving user interactions and opening up countless applications. Among them, haptic devices have mainly been used as training tools for surgeons. Indeed prior to surgery, under the assumption of known geometry and mechanical properties of the patient's organ, a surgeon would be able to plan and better choose between specific surgical paths/approaches. In this work, we used the 3D Systems Touch Haptic Device robot coupled with the SOFA plugin Geomagic to allow interactions between the instrument and the simulations. For this hypothetical simulation, we virtually simulate the contact between a surgical tool and a liver during surgery. The liver was described by an anisotropic Holzapfel Ogden model [Holzapfel et Ogden (2009); Pezzuto et al. (2014)], with an existing FEniCS implementation [Hauseux et al. (2018)] validated using the manufactured solution. Therefore, to assess the soundness of our implementation, we conducted a mesh convergence analysis on a beam and liver meshes provided in figure 4.5. We prescribed natural Dirichlet boundary conditions on one side and Neumann boundary conditions on the other side to obtain noticeable deformations. Such material could be described using the following strain energy density function

$$\psi = \psi_{\text{iso}}(\boldsymbol{F}) + \psi_{\text{vol}}(J). \tag{4.16}$$

ψ_{iso} and ψ_{vol} are the isochoric and volumetric part of the strain energy density function respectively. The volumetric part can be evaluated as a function of the bulk modulus κ of the material and J

$$\psi_{\text{vol}}(J) = \frac{\kappa}{4}(J^2 - 1 - 2\ln(J)), \tag{4.17}$$

$$\psi_{\text{iso}}(\boldsymbol{F}) = \frac{a}{2b}\exp\left[b\left(\mathrm{I}_1 - 3\right)\right] + \sum_{i=f,s}\frac{a_i}{2b_i}\exp\left[b_i\left(\mathrm{I}_{4i} - 1\right)^2\right] + \frac{a_{\text{fs}}}{2b_{\text{fs}}}\left(\exp\left[b_{\text{fs}}\mathrm{I}_{8\text{fs}}^2\right] - 1\right), \tag{4.18}$$

with

$$\mathrm{I}_{4f} = \boldsymbol{f_0}\cdot\boldsymbol{C}\cdot\boldsymbol{f_0},\ \mathrm{I}_{4s} = \boldsymbol{s_0}\cdot\boldsymbol{C}\cdot\boldsymbol{s_0},\ \text{and}\ \mathrm{I}_{8\text{fs}} = \boldsymbol{f_0}\cdot\boldsymbol{C}\cdot\boldsymbol{s_0}. \tag{4.19}$$

The transversely isotropic behavior can be obtained by removing the parameters a_{fs}, b_{fs}, a_s and b_s, while the isotropic behavior is obtained by also suppressing the two parameters a_f and b_f. This kind of model is frequently used to model orthotropic materials

(e.g. muscle with fibers or tendons). Vectors f_0, s_0 are the unit base vectors normal to the planes of symmetry. For our application, we selected the following material properties allowing to obtain sufficient deformation of the objects: $\kappa = 10^2$ MPa, $a = 1.10^2$ kPa, $b = 5$ Pa, $a_f = 16$ kPa, $b_f = 12.8$ Pa, $a_s = 18$ kPa, $b_s = 10$ Pa, $a_{fs} = 9$ kPa, $b_{fs} = 12$ Pa.

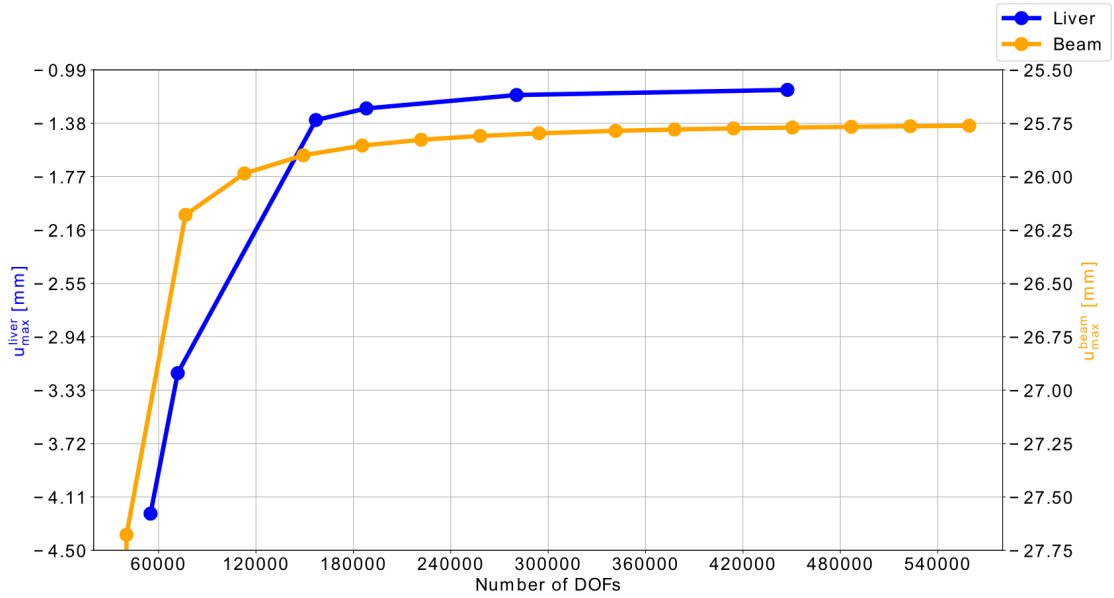


Figure 4.5: Plot of the mesh convergence analysis applied to a beam and liver using the Holzapfel and Ogden anisotropic material models with the parameters $\kappa = 10^2$ MPa, $a = 1.10^2$ kPa, $b = 5$ Pa, $a_f = 16$ kPa, $b_f = 12.8$ Pa, $a_s = 18$ kPa, $b_s = 10$ Pa, $a_{fs} = 9$ kPa, $b_{fs} = 12$ Pa. The maximum displacement of a beam $u_{\text{max}}^{\text{beam}}$ and liver $u_{\text{max}}^{\text{liver}}$ meshes are respectively computed in orange and blue for different mesh sizes increasing the number of DOFs.

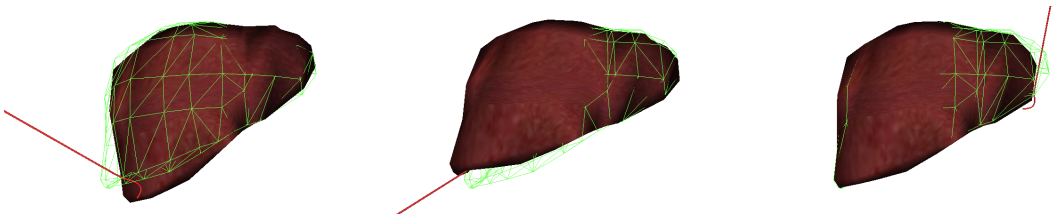


Figure 4.6: Three different deformation states of a liver in contact with a surgical tool connected to a haptic device. The surgical tool (in red) is guided by the user through the 3D Systems Touch Haptic Device to deform the liver from the initial configuration (green wire-frame) to a deformed state (textured). The liver is modeled using the Holzapfel Ogden anisotropic material with the following parameters $\kappa = 10^2$ MPa, $a = 1.10^2$ kPa, $b = 5$ Pa, $a_f = 16$ kPa, $b_f = 12.8$ Pa, $a_s = 18$ kPa, $b_s = 10$ Pa, $a_{fs} = 9$ kPa, $b_{fs} = 12$ Pa. The maximum displacement of a beam $u_{\text{max}}^{\text{beam}}$ and liver $u_{\text{max}}^{\text{liver}}$. In the case of contact detection, the contact forces are transmitted to the user through the haptic device. A video of the simulation is available as supplementary materials.

The instrument is assumed to be a rigid body kinematically constrained by the haptic

device position at each time step. The contact forces generated by the collision between the instrument and the liver are calculated using frictional contact and an implicit Euler scheme to solve the dynamic system [Courtecuisse et al. (2013)]. Finally, the contact forces are transmitted back to the user's hand through the haptic device. The result is a simulation running at 100 FPS (Frames Per Seconds) on average displayed in figure 4.6. The real-time performance has been obtained by using a small number of DOFs (543).

4.5 Discussion

We show that the SOniCS plugin is an efficient implementation of material models for hyperelastic simulations and enables the user to develop an intuitive understanding of the impact of modeling choices on the accuracy and reliability of the predictions. We first demonstrated a convergence study for Saint Venant-Kirchhoff material using P1 and P2 elements with the manufactured solution in section 4.3.2. Indeed, as expected, by refining the mesh, the L^2 relative errors for strain and displacement fields almost follow the theoretical slopes when increasing the number of DOFs (on a log-log plot), showing the stability of our method.

Then, from section 4.3.2, two main results are noteworthy from tables 4.1 and 4.2. First, the relative errors of the displacement and strain between SOniCS and SOFA, for both material models, are close to machine precision for P1, P2, Q1, and Q2 discretizations. Secondly, the last comment concerns the mean NR iteration time. We observe that the SOniCS implementation is slightly faster than SOFA for any elements using Saint Venant-Kirchhoff or Neo-Hookean material model. The reason for this difference is the need for SOFA to compute the shape functions and derivatives, then calculate the local residual and Jacobian using multiple tensor operations. Conversely, SOniCS has all those operations efficiently hard-coded in C kernels, thus performing faster than SOFA.

We compared the SOniCS and FEBio simulations for several material models: Saint Venant-Kirchhoff, Neo-Hookean, and Mooney-Rivlin in section 4.3.3. For P1 elements, the errors are close to machine precision for all 3 models. For P2 and Q2 elements, the 3 models display similar errors that still represent a minor error (less than 0.08 and 0.1, respectively) which is of same magnitude between SOFA and FEBio, as well as FEniCS and FEBio. The reasons for those minor errors could be the difference in the implementation of the elements or in the choice of solver parameters. For Q1 elements, we reached an accuracy near machine precision for Saint Venant-Kirchhoff and Neo-Hookean. However, the relative error for displacement rose close to 11 for the Mooney-Rivlin model, while similar behavior is observed for the relative strain errors. To further understand this

divergence, we included a third open-source software AceGen [Korelc (2022)]. AceGen similarly uses an automatic code generation package for the symbolic generation of new finite elements. The cantilever beam scenario presented in section 4.3.3 was reproduced using AceGen under the same conditions and with an identical Q1 discretization of the domain. Using the same metric as defined in equation 4.9, the results are the following: $L_u^2(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{Acegen}}) = 3.33$ and $L_u^2(\boldsymbol{u}_{\texttt{FEBio}}, \boldsymbol{u}_{\texttt{Acegen}}) = 14.19$. Even if SOniCS and Acegen showed similar results, a more in-depth study would be needed to confirm the soundness of our solution and explain the differences between FEBio and SOniCS solutions. Despite using a finer mesh the error was slightly lower but still noticeable. Eventually, FEBio has shown difficulty in converging with trivial parameter sets or when increasing the number of DOFs. Thus, as shown in tables 4.3,4.4, and 4.5, on average, SOniCS is solving the equation system faster than FEBio. Several reasons could explain those differences, such as the number of quadrature points used, the implementation of the Newton-Raphson scheme, or the solver parameters.

Finally, we showed the capabilities of the SOniCS plugin in simulating complex material models such as the Holzapfel Ogden anisotropic model coupled with a haptic device in section 4.4. The material model was effortlessly implemented for several elements (P1, P2, Q1, and Q2) without needing any manual derivation or coding. A manufactured solution was used in [Hauseux et al. (2018)] to validate the FEniCS implementation, while figure 4.5 demonstrates a convergence of the solution for a beam and liver shape using the SOniCS implementation. Therefore, the absence of analytical solutions or experimental data only proves the convergence to a numerical solution that is not necessarily a ground truth. The final result is a real-time simulator functional for surgeons' training or any other biomechanics simulation replicating the behavior of a liver in contact with a surgical tool.

4.6 Conclusion

We performed several numerical experiments to develop intuitive understanding of new material models in SOFA using the SOniCS plugin. First, we validated the most common hyperelastic material models: Saint Venant-Kirchhoff and Neo-Hookean using a manufactured solution. Then, utilizing FEBio as a reference, we verified our implementation of a Mooney-Rivlin material model using P1, P2, Q1, and Q2 elements. The final application employed a haptic device to interact with an anisotropic Holzapfel Ogden liver model in real-time. The study used our SOniCS plugin to generate optimized C code for complex material models compatible with SOFA. On one side, we benefited from FEn-

iCS automatic differentiation and code generation capabilities to bypass the difficulties of deriving and implementing the consistent Jacobian in SOFA. On the user side, we implemented compatible and user-friendly SOFA Forcefields to use the FEniCS C kernels. A SOFA user can now easily define a new material model by specifying its strain energy function, element geometry or family, and the quadrature scheme and degree only in Python. We made the open-source code and all data and test cases available as supplementary material.

Despite its generic nature and potential to support a wide range of scenarios involving the implementation of advanced hyperelastic laws, in this thesis, we will primarily use the SOniCS framework to develop a biomechanical model of the liver. By focusing on hyperelastic material models tailored to the liver's complex mechanical properties, this model will play a crucial role in the registration process. While SOniCS is versatile for broader applications, its primary use in this work is to provide an accurate liver biomechanical model aiming to solve the registration problem.

CHAPTER 5

ULTRASOUND VOLUME RECONSTRUCTION

5.1 Introduction

Typically, ultrasound volumes are reconstructed by tracking a sequence of frames, positioning them in space based on their corresponding tracking data, and then interpolating between them. Electromagnetic tracking systems are often employed for this task. However, their use in laparoscopic surgery is constrained due to size limitations. These systems can also introduce complexity and increase the cost of the surgical setup, while their accuracy may be compromised by the presence of metallic instruments in the surgical environment. In this section, we address the challenge of reconstructing IVUS volumes without relying on any tracking systems. This volume reconstruction is essential in the context of this thesis, as it serves as the intraoperative data. Once segmented, the IVUS volume provides key intraoperative features critical for the feature-based registration method employed in later chapters. The work presented in this chapter led to the following publication:

- **S. El hadramy**, J. Verde, KP. Beaudet, N. Padoy, S. Cotin. "Trackerless Volume Reconstruction from Intraoperative Ultrasound Images" In: Greenspan, H., et al. *Medical Image Computing and Computer Assisted Intervention – MICCAI 2023*. Lecture Notes in Computer Science, vol 14229 [El hadramy et al. (2023a)].

Several approaches have been proposed to address trackerless ultrasound volume reconstruction. Physics-based methods have exploited speckle correlation models between different adjacent frames [Mercier et al. (2005); Mohamed et Vei Siang (2019); Mozaffari et Lee (2017)] to estimate their relative position. With the recent advances in deep learning, recent works have proposed to learn a higher-order nonlinear mapping between adjacent frames and their relative spatial transformation. Prevost et al. (2017) first demonstrated the effectiveness of a convolution neural network to learn the relative motion between a pair of US images. Xie et al. (2021) proposed a pyramid warping layer that exploits the optical flow features and the ultrasound features to reconstruct the volume. To enable a smooth 3D reconstruction, a case-wise correlation loss based on 3D CNN and Pearson correlation coefficient was proposed in [Guo et al. (2021); Xie et al. (2021)]. Li et al. (2022) leverages past and future frames to estimate the relative transformation between each sequence pair; they used the consistency loss proposed in Miura et al. (2021). Recent work [Luo et al. (2022); Ning et al. (2022)] proposed to exploit the acceleration and orientation of an inertial measurement unit (IMU) mounted on the probe to improve the reconstruction performance and reduce the drift error. Despite the success of these approaches, they still suffer significant cumulative drift errors and mainly focus on reconstructing volumes from translation motion, as they are interested in standard ultrasound probes. This

limits the use of their methods in the case of the IVUS, which mainly employs rotational motion, as explained in chapter 1.

Motivated by the weakness of the state-of-the-art methods when it comes to large rotational probe motions and the difficulty of integrating IMU and electromagnetic sensors in the case of minimally invasive procedures, we introduce a new method for pose estimation and volume reconstruction in the context of minimally invasive trackerless ultrasound imaging. Our method uses a Siamese architecture based on a Sequence to Vector(Seq2Vec) neural network that leverages ultrasound images and features computed with the optical flow to learn relative transformation between a pair of images. Our method improves upon previous solutions in terms of robustness and accuracy, particularly in the presence of rotational motion. Such motion is predominant in the context highlighted above and is the source of additional non-linearity in the pose estimation problem. To the best of our knowledge, this is the first work that provides a clinically sound and efficient 3D IVUS volume reconstruction during minimally invasive procedures.

5.2 Method

In this work, we assume that the IVUS probe does not induce any deformation on the organ of interest during the volume acquisition. This assumption is realistic due to the small size of the probe. In addition, the IVUS probe is inserted inside the organ, which makes it move with the organ when human respiration occurs. Let $I_0, I_1 ... I_{N-1}$ be a sequence of N frames. Our aim is to find the relative spatial transformation between each pair of frames I_i and I_j with $0 \leq i \leq j \leq N-1$. This transformation is denoted $T_{(i,j)}$ and is a six degrees of freedom vector representing three translations and three Euler angles. To achieve this goal, we propose a Siamese architecture that leverages the optical flow in the sequences in addition to the frames of interest in order to provide a mapping with the relative frames' spatial transformation. The overview of our method is presented in Fig A.2.

We consider a window of $2k+3$ frames from the complete sequence of length N, where $0 \leq k \leq \lfloor \frac{N-3}{2} \rfloor$ is a hyper-parameter that denotes the number of frames between two frames of interest. Our method predicts two relative transformations between the pairs of frames (I_1, I_{k+2}) and (I_{k+2}, I_{2k+3}). The input window is divided into two equal sequences of length $k+2$ sharing a common frame. Both deduced sequences are used to compute a sparse optical flow allowing to track the trajectory of M points. Then, Gaussian heatmaps are used to describe the motion of the M points in an image-like format (see section 5.2.2). Finally, a Siamese architecture based on two shared weights Seq2Vec

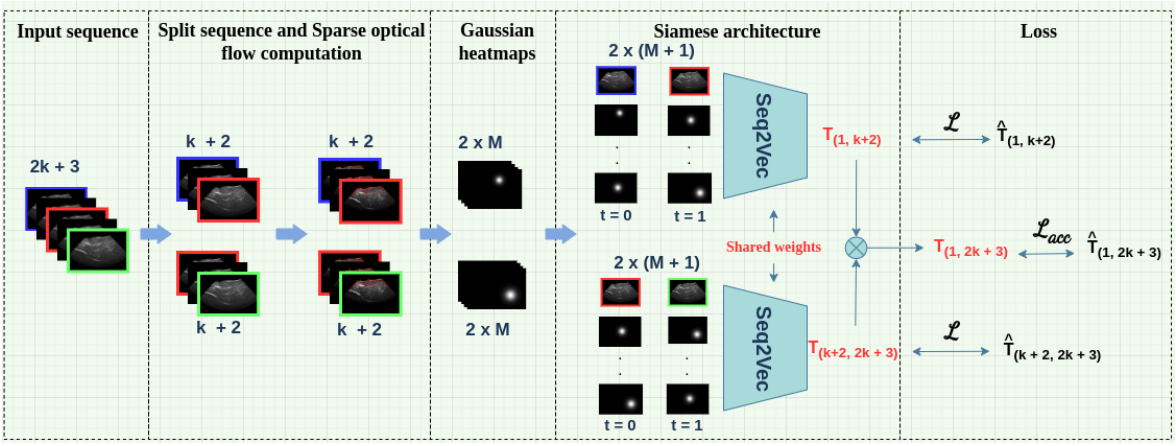


Figure 5.1: Overview of the proposed method. The input sequence is split into two equal sequences with a common frame. Both are used to compute a sparse optical flow. Gaussian heatmaps tracking M points are then combined with the first and last frame of each sequence to form the network's input. We use a Siamese architecture based on Sequence to Vector (Seq2Vec) network. The learning is done by minimising the mean square error between the output and ground truth transformations.

network takes as input the Gaussian heatmaps in addition to the first and last frames and predicts the relative transformations. In the following, we detail our pipeline.

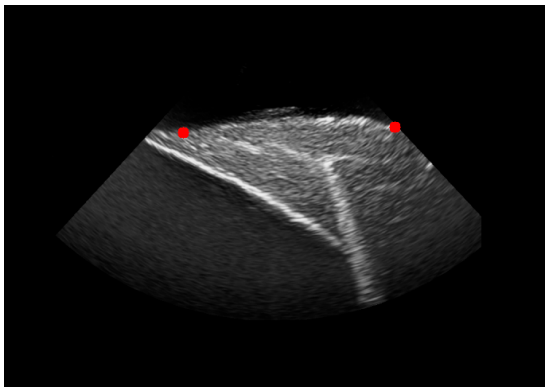
5.2.1 Sparse Optical Flow

Given a sequence of frames I_i and I_{i+k+1}, our aim is to find the trajectory of a set of points throughout the sequence. We choose the M most prominent points from the first frame using the feature selection algorithm proposed in Shi et Tomasi (1994). Points are then tracked throughout each pair of adjacent frames in the sequence by solving equation 5.1 which is known as the Optical flow equation. We use the pyramidal implementation of Lucas-Kanade method proposed in Bouguet (2001) to solve the equation. Thus, yielding a trajectory matrix $A \in \mathbb{R}^{M\times(k+2)\times 2}$ that contains the position of each point throughout the sequence. Figure 5.2 illustrates an example where we track two points in a sequence of frames.

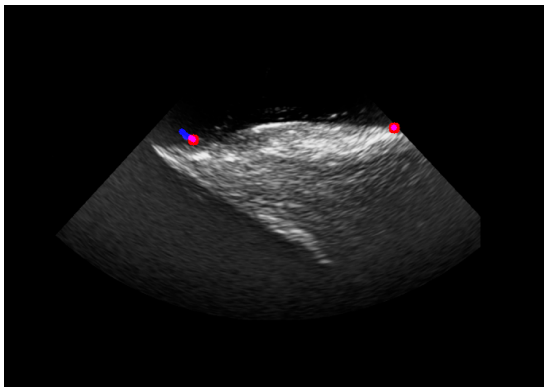
$$I_i(x, y, t) = I_i(x + dx, y + dy, t + dt) \tag{5.1}$$

5.2.2 Gaussian Heatmaps

After obtaining the trajectory of M points in the sequence $\{I_i | 1 \leq i \leq k+2\}$ we only keep the first and last position of each point, which corresponds to the positions in our frames of interest. We use Gaussian heatmaps $\mathcal{H} \in \mathbb{R}^{H\times W}$ with the same dimension as the ultra-



(a) First frame in the sequence



(b) Last frame in the sequence

Figure 5.2: Sparse Optical tracking of two points in a sequence, red points represent the chosen points to track, while the blue lines describe the trajectory of the points throughout the sequence.

sound frames to encode these points, they are more suitable as input for the convolutional networks. For a point with a position (x_0, y_0), the corresponding heatmap is defined in the equation 5.2.

$$\mathscr{H}(x,y) = \frac{1}{\sigma^2\sqrt{2\pi}} e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}} \tag{5.2}$$

Thus, each of our M points are converted to a pair of heatmaps that represent the position in the first and last frames of the ultrasound sequence. These pairs concatenated with the ultrasound first and last frames form the recurrent neural network sequential input of size $(M+1, H, W, 2)$, where $M+1$ is the number of channels (M heatmaps and one ultrasound frame), H and W are the height and width of the frames and finally 2 represents the temporal dimension.

5.2.3 Network Architecture

The Siamese architecture is based on a sequence-to-vector network. Our network maps a sequence of two images having $M+1$ channel each to six degrees of freedom vector (three translations and three rotation angles). The architecture of Seq2Vec is illustrated in the Fig 5.3. It contains five times the same block composed of two Convolutional LSTMs (ConvLSTM) [Shi et al. (2015)] followed by a Batch Normalisation. Their output is then flattened and mapped to six degrees of freedom vector through linear layers; ReLU is the chosen activation function for the first linear layer. We use an architecture similar to the one proposed in Shi et al. (2015) for the ConvLSTM layers. Seq2Vec networks share the same weights.

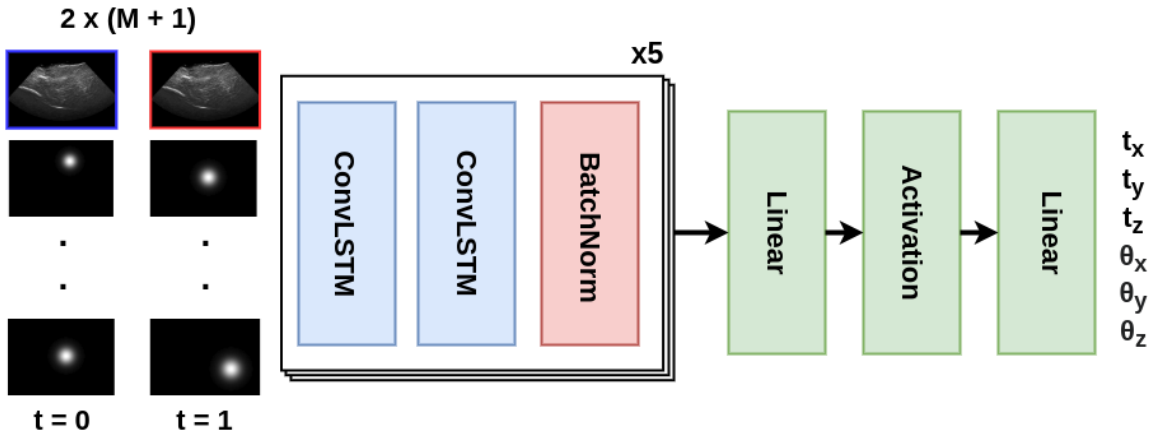


Figure 5.3: Architecture of Seq2Vec network. We use five blocks that contain each two ConvLSTM followed by Batch Normalisation. The output is flattened and mapped to a six degree-of-freedom translation and rotation angles through linear layers. The network takes as input a sequence of two images with $M+1$ channel each, M heatmaps and an ultrasound frame. The output corresponds to the relative transformation between the blue and red frames.

5.2.4 Loss function

In the training phase, given a sequence of $2k+3$ frames in addition to their ground truth transformations $\hat{T}_{(1,k+2)}$, $\hat{T}_{(k+2,2k+3)}$ and $\hat{T}_{(1,2k+3)}$, the Seq2Vec's weights are optimized by minimizing the loss function given in the equation 5.3. The loss contains two terms. The first represents the mean square error (MSE) between the estimated transformations ($T_{(1,k+2)}$, $T_{(k+2,2k+3)}$) at each corner point of the frames and their respective ground truth. The second term represents the accumulation loss that aims at reducing the error of the volume reconstruction. The effectiveness of the accumulation loss has been proven in Li et al. (2022). It is written as the MSE between the estimated $T_{(1,2k+3)} = T_{(k+2,2k+3)} \times T_{(1,k+2)}$ at the corner points of the frames and the ground truth $\hat{T}_{(1,2k+3)}$.

$$\mathscr{L} = ||T_{(1,k+2)} - \hat{T}_{(1,k+2)}||_2 + ||T_{(k+2,2k+3)} - \hat{T}_{(k+2,2k+3)}||_2 + ||T_{(1,2k+3)} - \hat{T}_{(1,2k+3)}||_2 \quad (5.3)$$

5.3 Experiments

5.3.1 Dataset acquisition and Implementation details

Six tracked sequences were acquired from an *ex vivo* swine liver to validate our method. A manually manipulated IVUS catheter was used (8 Fr lateral firing AcuNav™, 4–10 MHz)

connected to an ultrasound system (ACUSON S3000 HELX Touch, Siemens Healthineers, Germany), both commercially available. An electromagnetic tracking system (trakSTAR™, NDI, Canada) was used along with a 6 DOFs sensor (Model 130) embedded close to the tip of the catheter, and the PLUS toolkit [Lasso et al. (2014)] along with 3D Slicer [Fedorov et al. (2012)] was used to record the sequences. The frame size was initially 480 × 640. Frames were cropped to remove the patient and probe characteristics, then downsampled to a size of 128 × 128 with an image spacing of 0.22 mm per pixel. The first and end stages of the sequences were removed from the six acquired sequences, as they were considered largely stationary and aimed to avoid training bias. Clips were created by sliding a window of 7 frames (corresponding to a value of $k = 2$) with a stride of 1 over each continuous sequence, yielding a data set containing 13734 clips. The tracking was provided for each frame as a 4×4 transformation matrix. We have converted each to a vector of six degrees of freedom corresponding to three translations in mm and three Euler angles in $degrees$. For each clip, relative frame-to-frame transformations were computed for the frames number 0, 3, and 6. The distribution of the relative transformation between the frames in our clips is illustrated in the figure 5.4. It is clear that our data mostly contains rotations, in particular over the axis x. Heatmaps were calculated for two points ($M = 2$) and with a quality level of 0.1, a minimum distance of 7, and a block size of 7 for the optical flow algorithm (see Bouguet (2001) for more details). The number of heatmaps M and the frame jump k were experimentally chosen among 0, 2, 4, 6. The data was split into train, validation, and test sets by a ratio of 7:1.5:1.5. Our method is implemented in *Pytorch*[1] *1.8.2*, trained and evaluated on a *GeForce RTX 3090*. We use an Adam optimizer with a learning rate of 10^{-4}. The training process converges in 40 epochs with a batch size of 16. The model with the best performance on the validation data was selected and used for the testing.

5.3.2 Evaluation metrics and results

The test data was used to evaluate our method. It contains 2060 clips over which our method achieved a translation error of $\epsilon_{translation}$ of 0.449 ± 0.189 mm, and an orientation error of $\epsilon_{orientation}$ 1.3 ± 1.5 degrees. We have evaluated our reconstruction with a commonly used state-of-the-art metric called final drift error, which measures the distance between the center point of the final frame according to the real relative position and the estimated one in the sequence. On this basis, each of the following metrics was reported over the reconstructions of our method. **Final drift rate (FDR)**: the final drift divided by the sequence length. **Average drift rate (ADR)**: the average cumulative drift of all

[1] https://pytorch.org/docs/stable/index.html

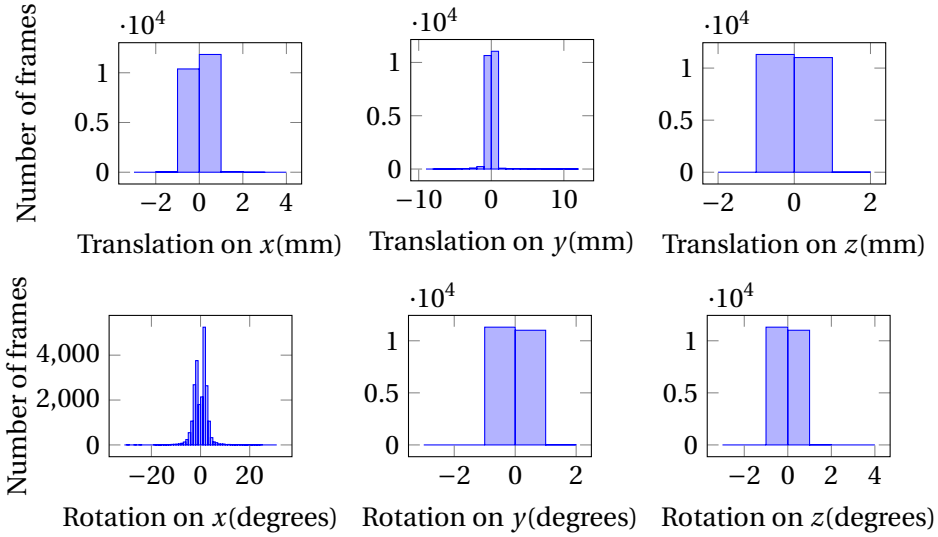


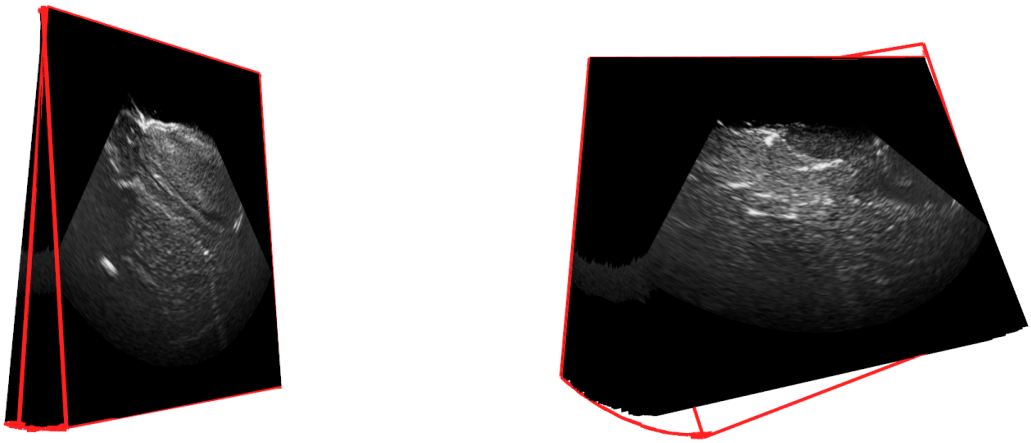
Figure 5.4: The distribution of the relative rotations and translations over the dataset

Models	**FDR(%)**	**ADR(%)**
CNN [Prevost et al. (2017)]	**31.88(15.76)**	**39.71(14.88)**
MoNet [Luo et al. (2022)]	**15.67(8.37)**	**25.08(9.34)**
Ours	**23.11(11.6)**	**28.71(12.97)**

Table 5.1: The mean and standard deviation FDR and ADR of our method compared with state-of-the-art models MoNet [Luo et al. (2022)] and CNN [Prevost et al. (2017)]

frames divided by the length from the frame to the starting point of the sequence. Table A.1 shows the evaluation of our method over these metrics compared to the state-of-the-art methods MoNet Luo et al. (2022) and CNN Prevost et al. (2017). Both state-of-the-art methods use IMU sensor data as additional input to estimate the relative transformation between two relative frames. Due to the difficulty of including an IMU sensor in our IVUS catheter, the results of both methods were reported from the MoNet paper, where the models were trained on arm scans. See Luo et al. (2022) for more details.

As the table A.1 shows, our method is comparable with state-of-the-art methods in terms of drift errors without using any IMU and with rotational probe motion as one may notice in our data distribution in the Fig 5.4. Fig 5.5 shows the volume reconstruction of two sequences of different sizes with our method in red against the ground truth slices. Despite the rotational motion of the probe, the relative pose estimation results obtained by our method remain accurate. However, one may notice that the drift error increases with respect to the sequence length. This remains a challenge for the community even in the case of linear probe motions.



(a) Sequence of length 50 frames

(b) Sequence of length 300 frames

Figure 5.5: The reconstruction of two sequences of lengths 50 and 300 respectively with our method in red compared with the ground truth sequences.

5.4 Conclusion

In this work, we proposed the first method for trackerless ultrasound volume reconstruction in the context of minimally invasive surgery. Our method does not use any additional sensor data and is based on a Siamese architecture that leverages the ultrasound image features and the optical flow to estimate relative transformations. Our method was evaluated on *ex vivo* porcine data and achieved translation and orientation errors of 0.449 ± 0.189 mm and 1.3 ± 1.5 degrees, respectively, with a fair drift error. In future work, we will extend our work to further improve the volume reconstruction and use it to register a pre-operative CT image to provide guidance during interventions.

CHAPTER 6

REAL-TIME VESSEL-GUIDED AUGMENTED REALITY

In the two previous chapters, we laid the foundation for the work presented in this chapter. In **Chapter 4**, we introduced SOniCS, a framework designed to develop advanced hyperelastic material models by integrating the computational power of FEniCS with the real-time simulation capabilities of SOFA. In our context, this framework simplifies the process of implementing complex material laws and is essential for creating liver biomechanical models. In **Chapter 5**, we developed a method for reconstructing a 3D volume from an IVUS sequence without tracking, allowing intraoperative visualization of internal liver structures, such as the portal and hepatic veins. This 3D volume serves as intraoperative data. In this chapter, we present the core contribution of this thesis: a method for real-time registration of the intraoperative IVUS volume with preoperative CT data. This is achieved using a neural network trained on data generated by the biomechanical liver model. The proposed method enables registration between the intraoperative and preoperative data. The work presented in this chapter led to the following publication:

- **S. El hadramy**, J. Verde, N. Padoy, S. Cotin. "Towards Real-Time Vessel Guided Augmented Reality for Liver Surgery" *2024 IEEE International Symposium on Biomedical Imaging (ISBI)*, Athens, Greece, 2024, pp. 1-5 [S. et al. (2024a)].

6.1 Introduction

As previously outlined in **Chapter 1**, numerous studies have demonstrated the benefits of incorporating AR into operating rooms to assist with liver surgeries [Haouchine et al. (2013)]. This AR relies on effective registration between preoperative and intraoperative data. This thesis considers IVUS as the intraoperative modality for several reasons discussed in **Chapter 1**. In this chapter, we propose a patient-specific non-rigid registration method that utilizes biomechanical models for their capability to extrapolate from sparse and noisy data, as noted earlier. Given a preoperative CT scan, we segment the liver and its internal structures to create a patient-specific digital twin using biomechanical models. This digital twin is used to simulate IVUS data by introducing deformations, removing branches, and adding noise to the vessel tree. Through this process, we generate a training and validation dataset. For the non-rigid registration, we employ a U-Net architecture to learn the mapping between the intraoperative vascular tree centerlines and the corresponding displacement field, enabling non-rigid registration between the preoperative model and the intraoperative vascular tree. Additionally, we leverage the graph-like structure of the vascular tree to match the main portal veins in both their preoperative and intraoperative configurations, thereby allowing for rigid registration. Details about the method are explained in the following sections. It is important to note that we validate

our method using simulated IVUS data due to the lack of available real IVUS datasets. This chapter is organized as follows: Section 6.2 details the method, including constructing the biomechanical model, the neural network responsible for non-rigid registration, and the approach for rigid registration. Section 6.3 presents the experimental results, demonstrating the efficacy of our proposed method. We conclude this work in section 6.4.

6.2 Method

6.2.1 Overview

The proposed method is patient-specific and works under the assumption that during the IVUS acquisition, the root portal vein (i.e., main trunk) and its branches are visible. This assumption is in line with the clinical workflow, as clinicians use such branches as anatomic landmarks to guide the intervention. The method is made of three essential steps (Figure 6.1). First, an initial registration to bring preoperative and intraoperative data onto the same reference. Then, we create a digital twin of the organ from the preoperative data. Finally, we train a neural network on the non-rigid registration task using data generated with the biomechanical model.

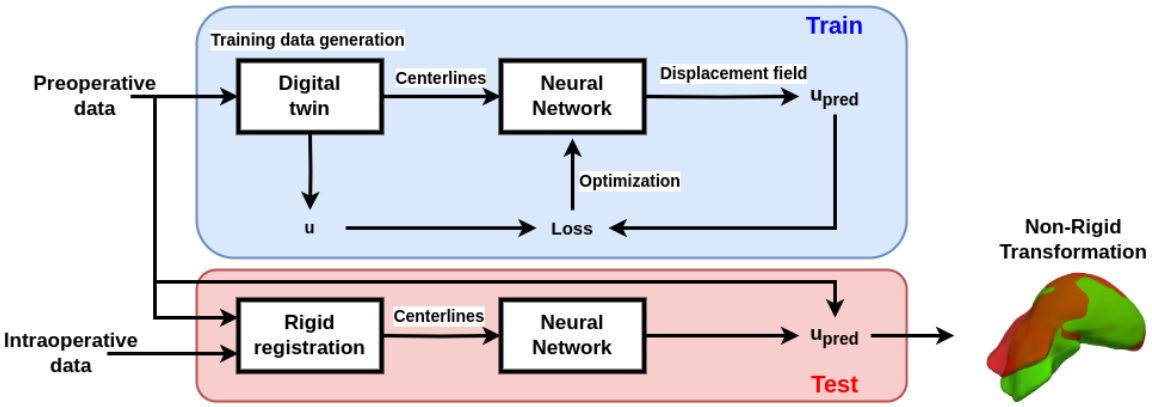


Figure 6.1: A liver digital twin is used to train a network on the non-rigid registration task. The initial registration is performed thanks to the graph-like structure of the vascular trees.

6.2.2 Digital Twin

6.2.2.1 Biomechanical model of the parenchyma

We formulate a boundary value problem to compute the deformation of an elastic material under both Dirichlet and Neumann boundary conditions. The geometry of the or-

gan, known *a priori*, occupies a volume Ω with boundary Γ. The Dirichlet and Neumann boundary conditions are on Γ_D and Γ_N, two subsets of Γ. See illustration in Figure 6.2. The elastic properties of soft tissues can be characterized using principles from continuum mechanics.

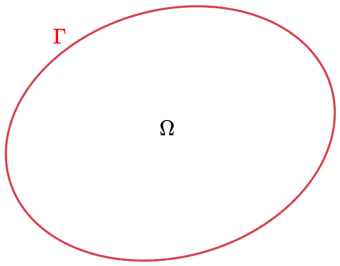


Figure 6.2: Illustration of the domain

By employing the *Lagrangian* formulation, the relationship between the deformed (x) and undeformed (X) states, at each point along the geometry can be expressed as

$$\mathrm{x} = \mathrm{X} + \mathrm{u} \tag{6.1}$$

Where u is the displacement field. The deformation gradient tensor $\mathbb{F} = \mathbb{I} + \nabla_{\mathrm{X}} \boldsymbol{u}$, provides a local description of the deformation and the Green-Lagrange strain tensor $\mathrm{E} \in \mathbb{R}^{3\times3}$ as expressed following equation 6.2

$$\mathrm{E} = \frac{1}{2}(\mathrm{C} - \mathrm{I}) \tag{6.2}$$

Where $\boldsymbol{C} = \mathbb{F}^T\mathbb{F}$ is called the right Cauchy-Green deformation tensor and $\boldsymbol{I}$ the identity matrix. Hyperelastic material is often employed to describe the behavior of soft tissues that undergo large deformations. According to the chosen hyperelastic material, the strain-energy density function, noted W, can be expressed using a set of parameters that describe the material stiffness. The stress-strain relationship, also known as constitutive law, is obtained by differentiating W with respect to C, see equation 6.3

$$\mathrm{S} = 2\frac{\partial \mathrm{W}}{\partial \mathrm{C}} \tag{6.3}$$

With S being the second Piola-Kirchhoff stress tensor. The boundary value problem is then formulated as in equation 6.4, where b represents the body forces, n the unit normal to Γ_N, and t the traction forces applied on Γ_N domain.

$$\begin{cases} \nabla(FS) = b \quad \text{on} \quad \Omega \\ u(X) = 0 \quad \text{on} \quad \Gamma_D \\ (FS) \cdot n = t \quad \text{on} \quad \Gamma_N \end{cases} \tag{6.4}$$

The weak form of Equation 6.4, brings forward the boundary term and is expressed following Equation 6.5. Readers are invited to refer to chapter 3 for more details.

$$\int_\Omega (\boldsymbol{FS}) : \delta \boldsymbol{E} \, d\Omega = \int_\Omega \boldsymbol{b\eta} \, \boldsymbol{d\Omega} + \int_{\Gamma_N} \boldsymbol{t\eta} \, \boldsymbol{d\Gamma} \tag{6.5}$$

With $\boldsymbol{\delta E} = \frac{1}{2}(\boldsymbol{F^T \nabla \eta} \, \boldsymbol{\nabla^T \eta F})$ being the variation of the strain, and $\boldsymbol{\eta} = \{\boldsymbol{\eta} \in \boldsymbol{H^1(\Omega)} \mid \eta = 0 \ on \ \Gamma_D\}$ is any vector-valued test function in an Hilbert space $\boldsymbol{H^1(\Omega)}$. In this work, we choose Saint-Venant-Kirchhoff material for the parenchyma model. In addition, Γ_D corresponds to the intersection between the parenchyma and the portal vein (Fig 6.4.)

6.2.2.2 Finite Element solution

For a given set of loads, which could include traction forces t and body force b, the equation 6.5 is solved using a finite element simulation to retrieve the relative displacement field u, based on a set of patient-specific parameters λ that describe the material properties or the location of Dirichlet boundary conditions. The domain Ω is discretized either using triangle elements or with an Immersed Boundary method (IBM) [Brunet (2020)], as shown in the left and right images of Figure 6.3 respectively. Using an IBM on a regular grid makes it compatible with the inputs and outputs of CNN architectures (See section 7.2.2). Given the equation's nonlinearity, a nonlinear system of equations must be solved to approximate the unknown displacement. Starting from an initial displacement u^0, an iterative Newton-Raphson method is employed to find a correction δ_u^n after n iterations, which satisfies the linearized set of equations, see Equation 6.6.

$$\dot{K}^{n-1} \delta_u^n = r(u^0 + \delta_u^{n-1}) + b \tag{6.6}$$

In this context, $\dot{K}$ represents the tangent stiffness matrix, while $\boldsymbol{r}$ denotes the internal elastic force vector. At each iteration, it is necessary to compute both $\dot{K}$ and $\boldsymbol{r}$ and solve the resulting linear system. The Newton-Raphson method converges effectively only when the initial displacement $\boldsymbol{u^0}$ is close to the actual solution. Therefore, large loads must be applied incrementally in small steps, which can necessitate a significant number of iterations to achieve convergence.

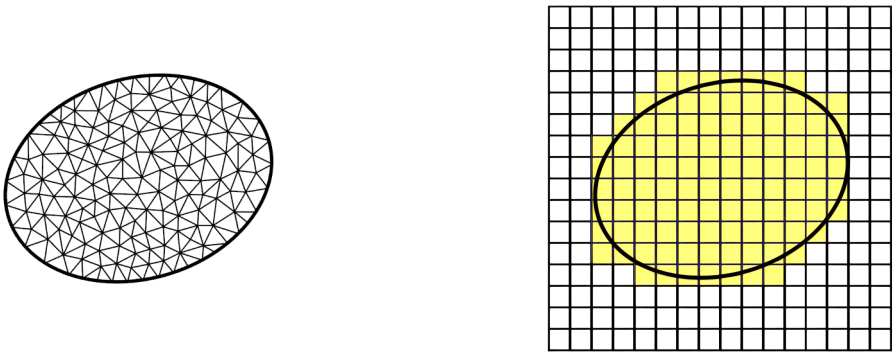


Figure 6.3: Left: Discretization of the domain Ω using triangle elements. **Right**: Discretization of the domain Ω using an Immersed Boundary Method, the geometry is discretized using regular elements, and the yellow cells are selected as part of the domain.

Figure 6.4 illustrates an example of a discretization of the parenchyma domain Ω with boundaries Γ_N and Γ_D.

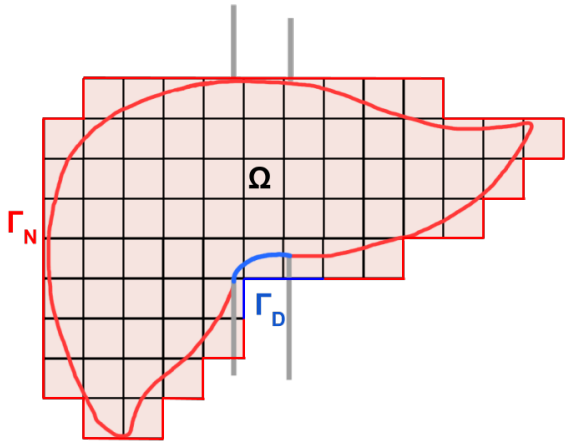


Figure 6.4: Formulation of the problem. The parenchyma occupies a domain Ω, We consider both Dirichlet and Neuman boundaries Γ_D and Γ_N respectively. The domain Ω is discretized using hexahedron elements.

6.2.2.3 Internal Structures mapping

Now that we have established a biomechanical model of the parenchyma, we constrain the internal structures, such as the vessel trees, to this parenchyma model. This is done by interpolating the displacement field of the parenchyma model, allowing us to compute the relative displacement field for the internal structures. By applying this interpolation, we ensure that the movement of the vessel trees and other internal elements remains consistent with the overall deformation of the parenchyma, maintaining accurate spatial relationships within the tissue model. The parenchyma model acting as a master imposes its displacements to the slaves, i.e, internal structures. We use a linear operator between

the slave (x_s) and the master (x_m) discretization, where J is the Jacobian matrix written as follows:

$$J = \frac{\partial x_s}{\partial x_m} \tag{6.7}$$

6.2.3 Initial registration

We recall that the root portal vein is assumed to be visible in the intraoperative data. Given two vascular trees, we identify and match the root branch of both. This allows for the first global registration by matching those branches. Then, the affine registration is performed through an ICP [Besl et McKay (1992)] algorithm between both vessel trees surface points. The global registration ensures a precise and fast convergence of the ICP, which is known to be a local registration method and rely on a rough alignment as initialization. The identification of the root branch is done by considering a graph structure originating from the portal vein. The graph is constructed from the vessel tree centerlines, where nodes and edges correspond respectively to bifurcations and branches in the tree. From an arbitrary orientation of the graph, we identify branches with either no parent or no children in the oriented graph. The root branch of the portal vein is determined as the one with the highest radius. This strategy aligns with the gross anatomy of the portal vein, and its branching pattern. We demonstrate in the results section that this approach is robust to partial and noisy intraoperative vessel trees.

6.2.4 Non-rigid registration

For an intraoperative IVUS volume, we perform segmentation of the portal vein to extract its structure from the surrounding tissues. Once the portal vein is segmented, we compute its centerlines, which provide a simplified representation of the vein's geometric path. These centerlines are crucial for subsequent processing, as they serve as a key feature for aligning the intraoperative data with preoperative imaging during registration tasks. Given these centerlines, in this section, we aim to update the preoperative data in order to match the intraoperative liver state. For the sake of clarity, in the following, let p_p be the preoperative parenchyma, p_i the intraoperative parenchyma and c_i the intraoperative vessels centerlines. We aim to find a function f that minimizes the following objective function:

$$\mathscr{J} = ||p_p \circ f(c_i) - p_i||_2^2 \tag{6.8}$$

In the equation 6.8, given a set of points c_i, representing the centrelines of the intraoperative vessel tree, f outputs a displacement field that allows a non-rigid registration

between p_p and p_i discretized domains. We approximate f using the U-Net architecture, similar to the one proposed by Brunet et al. (2019). The neural network takes as input a regular grid in which we encode the intraoperative centerlines. This grid has a high resolution and encodes the position of the closest points of the centerlines at each of its nodes. The output of the neural network, which describes the displacement field, has the same shape as the input.

The training set is synthetically generated from the biomechanical model. Each sample of the training set corresponds to a deformed state of the model. This deformation is obtained by applying a set of forces at several local locations of the liver surface. The forces are sampled from a prior distribution. The locations are chosen by considering random points on the liver surface and a sphere of radius r for each point. The intersections between these spheres and the liver surface are the local locations where the forces are applied. Figure 6.5 illustrates this process: the rest state of the liver is in gray, the blue spheres are the locations where the forces are applied, and the red parenchyma is the resulting deformation.

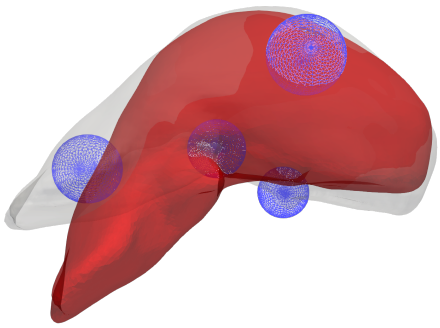


Figure 6.5: Example of a training set generation sample. The spheres (in blue) centers are randomly chosen on the surface's rest state (in gray). Forces are applied at their intersection with the surface. The resulting deformation is shown in red.

Once the parenchyma is deformed, the linear interpolation described in section 6.2.2.3 is employed to get the relative vessel trees' deformation. We use the method proposed by Antiga et al. (2003) to extract the centerlines of these vessel trees. The process of generating deformations of the liver parenchyma and the vessel tree using the biomechanical model is iterated multiple times to create a robust dataset with realistic deformations. For each sample in this dataset, we treat the deformed vessel tree as if it were intraoperative data derived from an IVUS volume. To simulate a realistic IVUS segmentation, we further process the deformed vessel tree by removing some branches and introducing noise. This simulated IVUS segmentation is then used to compute the centerlines, which are the input of the neural network.

During training, the network predicts the relative displacement field, for which we have a ground truth, as the deformation was generated by the biomechanical model. The network is trained to minimize the MSE between the predicted displacement fields and the ground truth, ensuring that the model learns to accurately predict the deformations that align intraoperative data with preoperative imaging.

6.3 Experiments and results

6.3.1 Dataset and implementation details

To assess our method, the liver parenchyma, portal vein, and tumor are segmented from real patient data from Soler et al. (2010). We build a biomechanical model considering 4000 Pa for the Young modulus and 0.4 for the Poisson's ratio. We then generate training and validation datasets. The applied forces are sampled from a uniform distribution over $[0,30]$ N, to ensure we cover both small and large organ deformations. The generated dataset contains 3600 different deformations, 3200 are used for training and 400 for validation. We consider two cases related to the IVUS acquisition. First, when the full vascular tree is observed during the acquisition, this corresponds to the case where the IVUS catheter was rotated 360°. The second scenario corresponds to the case where only a partial vascular tree was observed during the IVUS acquisition. We recall that in both cases, we assume the main portal vein is observed as it serves for the initial rigid registration. The following subsections describe the validation of our method in both cases. We use the biomechanical model as a baseline and validate our networks' prediction using target registration error (TRE) metric on the tumors' localization. The biomechanical model is implemented through SOFA Framework [Faure et al. (2012)] with the SOniCS plugin presented in Chapter 4. The neural network is implemented with PyTorch [1].

6.3.2 Full vascular tree

In this case, we consider a scenario in which the full intraoperative vascular tree can be observed with a 360° rotation of the IVUS catheter. The network was trained on the 3200 training samples during 100 epochs with a batch size of 16 and tested on the 400 validation samples. The results of the network of the validation set are presented in table 6.1. The target registration error is computed over the tumor's location inside the parenchyma. We achieve a mean target registration error of 0.6 mm with a maximum error of 2.97 mm over

[1] https://pytorch.org/docs/stable/index.html

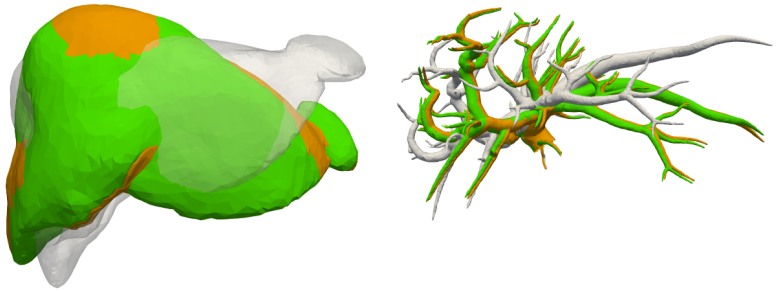


Figure 6.6: Visualization of a validation sample with a full intraoperative vascular tree. In gray is the rest state of the liver, corresponding to the preoperative liver shape. In green, the ground truth deformation is generated with the biomechanical model. In orange, the deformation is predicted by the neural network.

	$e_{mean}(mm)$	$e_{max}(mm)$	$e^{relative}_{mean}(\%)$	$e^{relative}_{max}(\%)$
U-Mesh [Brunet et al. (2019)]	0.83 ± 0.42	3.01	0.22 ± 0.11	0.78
Ours (30%)	0.59 ± 0.37	2.36	0.16 ± 0.10	0.63
Ours (50%)	0.63 ± 0.24	1.28	0.17 ± 0.06	0.33
Ours (full)	0.60 ± 0.37	2.97	0.16 ± 0.09	0.81

Table 6.1: Results of our method in terms of TREs for the scenarios: Full, Partial (50%) and Partial (30%) vessel tree. The errors are calculated by comparing the network's prediction with the biomechanical model deformation for the same intraoperative deformation.

the 400 validation samples. We also report the relative errors, which correspond to a percentage of the deformation magnitude for each sample. The predictions are computed in 34 ms. Figure 6.6 shows the results of a validation sample, where the matching between prediction and ground truth can be observed. In gray is the rest state of the organ (preoperative state); in green is the ground truth deformation (from the biomechanical model); and in orange is the prediction of the network.

Moreover, we have compared our results to the U-Mesh approach proposed by Brunet et al. (2019). Their intraoperative modality is laparoscopic video, where a 3D point cloud on the organ's surface is reconstructed from the video frames. We used their implementation to synthesize the point clouds and train and validate the network; the results are also shown in the table 6.1.

6.3.3 Partial vascular tree

Using the same process as the previous scenario, we have generated a dataset that corresponds to a scenario in which only a partial vascular tree is visible during the IVUS acquisition. In this scenario, we have considered two cases where 50% and 30% of the vascular

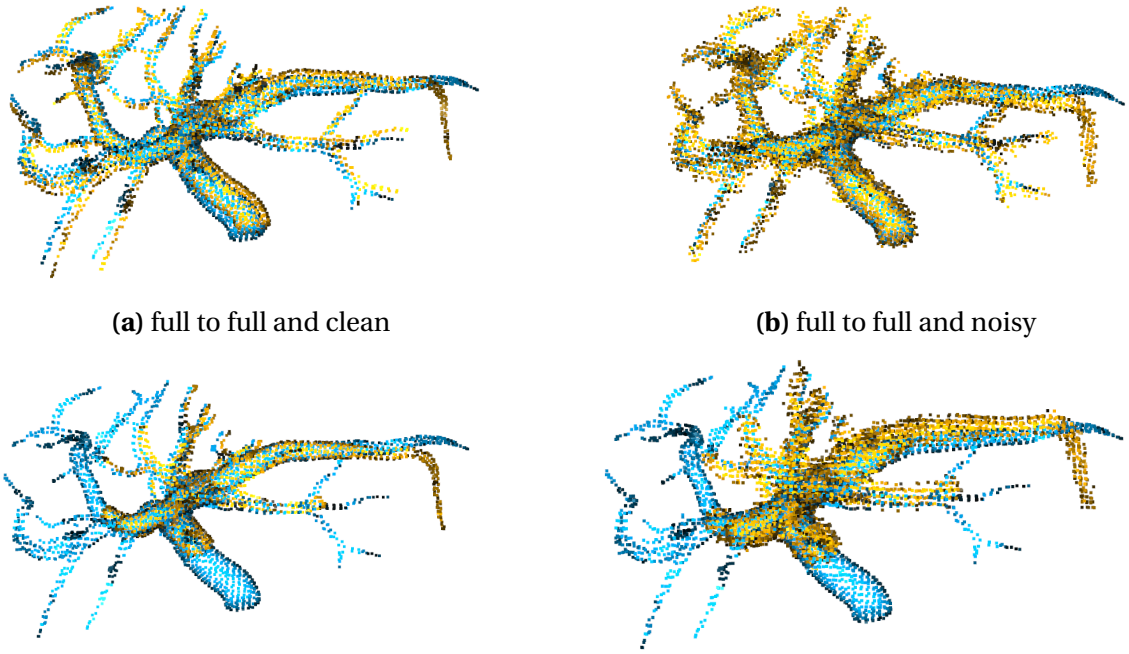
tree are observed, respectively. In both cases, the vascular branches are chosen in the areas where the tumors are located. This is because, surgeons know in which anatomical segment of the liver the tumors are located and usually acquire data in these areas. A dataset is generated for each of these scenarios and the network was trained on the 3200 training samples during 100 epochs with a batch size of 16. The results of the network on the validation dataset of size 400 samples are reported in Table 6.1 for both cases. Results show that the network still performs in terms of TRE at the tumors' locations. With a mean TRE of 0.59 mm and 0.63 mm for 50% and 30% intraoperative vascular tree, respectively. With a maximum TRE of 1.28 mm and 2.36 mm over the 400 validation samples. The computation time is 34 ms.

6.3.4 Initial registration

In this section, we present qualitative results of the initial rigid registration between the preoperative vascular tree and various simulated scenarios of segmented intraoperative IVUS data. Let v_p and v_i be two vessel trees surface points, located in the same reference frame, with a non-rigid deformation between them. A random rigid transformation is applied to v_p to obtain a third vessel tree surface points, denoted v_p^T. Following our initial registration method explained in section 6.2.3, we register v_p^T to v_i and compare the obtained result to v_p. To simulate different IVUS scenarios, we consider four cases of v_i, denoted respectively "Full and clean", "full and noisy", "partial and clean" and "partial and noisy". Figure 6.7 illustrates the results of our initial registration approach in each of these cases. Our experiments show that the initial registration is robust to intraoperative partial/full and/or clean/noisy vascular trees.

6.3.5 Segmentation sensibility

The accuracy of centerlines is highly dependent on the quality of the segmentation from intraoperative ultrasound images. Despite recent advances in medical image segmentation, ultrasound-based vascular segmentation remains particularly challenging due to the noisy and low-contrast nature of ultrasound data. It is often highly dependent on the operator or the software used for segmentation. To account for this variability, we introduced random noise with a maximum magnitude of 1 mm along the x, y, and z directions during network training, simulating segmentation uncertainty. We then tested the network using centerlines with higher noise levels of 1.5 mm and 2 mm, which were not encountered during training. Fig 6.8 shows the results on partial intraoperative vascular trees, and as demonstrated, the variability introduced during training enabled the network to remain robust even to these unseen noise levels, with a mean TRE below 2



(a) full to full and clean

(b) full to full and noisy

(c) full to partial and clean

(d) full to partial and noisy

Figure 6.7: Results of the initial registration.

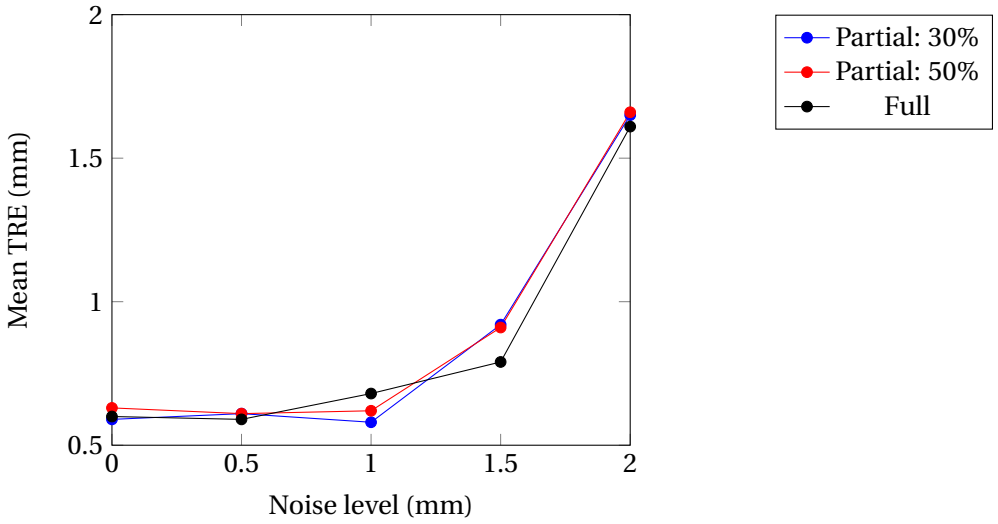


Figure 6.8: Segmentation sensibility results for full, 30% and 50% partial intraoperative vascular trees

mm across all noise scenarios. This highlights the robustness of our method in handling segmentation errors from ultrasound imaging.

6.4 Conclusion

In this work, we proposed a method for real-time non-rigid registration of preoperative CT data with simulated intraoperative IVUS data. The method was trained on synthetically generated datasets from real human cases and validated against the FEM solution of a biomechanical liver model. The intraoperative data were simulated from CT segmentations by considering partial vessel trees and adding noise to replicate the challenges associated with IVUS segmentation. Our approach demonstrated strong performance, achieving a mean and maximum TRE of less than 3 mm across 400 synthetic samples. Additionally, our method outperformed U-Mesh on synthetic data. U-Mesh uses laparoscopic camera views of the surface as the intraoperative modality which makes the initial rigid registration challenging, while our method leverages the graph-like structure of the intraoperative vascular tree, allowing for both rigid and non-rigid registration. Although we did not validate our method with real intraoperative IVUS data due to a lack of available datasets, the results highlight the robustness of our approach, particularly in handling partial and noisy data. This, combined with our superior performance on synthetic data, makes our method highly promising for future validation with real intraoperative IVUS data.

CHAPTER 7

MODEL PARAMETRIZATION

In the previous chapter, we presented a patient-specific registration method that leverages U-Mesh to align intraoperative centerlines with the preoperative liver model. The network was trained using data generated from a biomechanical model of the liver. However, when constructing this biomechanical model, we assumed knowledge of patient-specific characteristics such as material stiffness and organ boundary conditions. These assumptions are strong since these quantities are often difficult to estimate from preoperative data. In this chapter, we extend the U-Mesh framework so that it can predict displacement for a range of patient-specific characteristics, including material stiffness and boundary conditions. Moreover, we show that by leveraging this extended version of U-Mesh, we can estimate these patient-specific characteristics in real-time from intraoperative observations. This is achieved through an optimization loop, which iteratively refines the material properties and boundary conditions based on the observed data, ensuring that the model dynamically adapts to the individual patient's anatomical and mechanical variations during the procedure. The work presented in this chapter led to the following publications:

- **S. El hadramy**, N. Padoy, S. Cotin. "HyperU-Mesh: Real-time deformation of soft-tissues across variable patient-specific parameters". Computational Biomechanics Workshop at MICCAI 2024. [S. et al. (2024b)]

- **S. El hadramy**, B. Acidi, N. Padoy, S. Cotin. "Optimization in latent space for real-time intraoperative characterization of digital twins". *Under review*

7.1 Introduction

Digital twins can play a major role in healthcare by leveraging real-time data integration and virtual simulations to enhance patient care, enable predictive analytics, optimize medical devices, and facilitate planning and intraoperative guidance [Servin et al. (2024)]. Several studies have demonstrated the benefits of physics-based digital twins (PBDTs) [Brunet (2020); Haouchine et al. (2013)] for their accuracy and predictive ability over sparse and noisy data. PBDTs rely on computational models that simulate the behavior of physical systems using fundamental principles from physics (e.g., elasticity, thermodynamics), where the finite element method (FEM) is often employed to solve the underlying partial differential equations (PDEs) [Suwelack et al. (2014)]. It is essential to prioritize their accuracy and real-time responsiveness to fully benefit from the advantages of physics-based digital twins in computer-assisted interventions. Accurate models ensure the simulations reflect true-to-life scenarios, leading to more reliable and effective

outcomes. Meanwhile, real-time capabilities allow these digital twins to provide immediate feedback and adjustments during interventions, enhancing their practical utility in dynamic and critical environments.

7.1.1 Accuracy

The accuracy of PBDTs relies on patient-specific characteristics. These characteristics, which are model-dependent, include a range of properties such as density, stiffness, thermal conductivity, and other parameters essential for simulating the behavior of an organ or system [Servin et al. (2024)]. Additionally, boundary conditions (BCs) are critical for the accuracy of PBDTs, as they define the interactions between the organ and its environment, significantly influencing the simulation outcomes. Despite their importance, obtaining precise measurements of these model parameters is challenging. Traditional imaging and other non-invasive techniques often fall short of accurately capturing the necessary details of these properties. For instance, properties like tissue stiffness or thermal conductivity are not readily measurable through standard imaging modalities. Similarly, accurately defining boundary conditions involving complex interactions between tissues and their surrounding environment is often infeasible without invasive procedures.

7.1.1.1 Material properties

As outlined in chapter 3, hyperelastic formulations are commonly employed in the literature to describe the behavior of soft tissues that experience deformations. St. Venant-Kirchhoff, Neo-Hookean, and Mooney-Rivlin are among Hyperelastic models that have been used to characterize soft-tissue. Traditionally, parameter sets for various models are determined through uniaxial tests, where researchers identify the parameters that align with the experimentally observed stress-strain relationship according to their strain energy model [Veronda et Westmann (1970)]. This method, while effective, can be limited by its reliance on straightforward experimental setups. To address these limitations, a more advanced approach involving iterative parameter identification using inverse finite element analysis has been proposed [Boonvisut et Cavuşoğlu (2013)]. This method enhances accuracy by refining parameter estimates through successive approximations. For instance, [Mehrabian et Samani (2009)] utilized this technique to estimate tissue parameters modeled with the Veronda-Westmann model. Additionally, [Han et al. (2011)] applied this approach to develop a patient-specific biomechanical model of the breast, demonstrating its potential for creating highly personalized and accurate tissue models.

However, the iterative nature of these methods means that numerous simulations must be performed to converge on a precise set of patient-specific characteristics. Each simulation can be computationally intensive, especially when modeling complex, nonlinear behaviors typical of soft tissues. Consequently, the time required to reach a solution can hinder the practical application of these methods in dynamic and time-sensitive scenarios, such as surgical procedures or real-time diagnostics.

7.1.1.2 Boundary conditions

BCs are crucial for the accuracy of PBDTs as they define the interactions between an organ and its surrounding environment. In the literature, zero-displacement boundary conditions are often characterized using priors. For instance, in the case of the liver, the points of attachment with the ligaments are usually considered [Brunet (2020)]. However, only a few studies have tackled the challenge of estimating these conditions from intraoperative data. Plantefeve et al. (2016) used a statistical atlas to estimate liver-ligament connectivity, but their method lacks robustness due to inter-patient variations. Another study estimated BCs by registering two preoperative scans under different deformations. While effective, this approach does not align with clinical settings or real-time requirements [Peterlik et al. (2014)]. Tagliabue et al. (2021) proposed a pipeline that estimates BCs from intraoperative point clouds of the visible surface. Their method utilizes nearest-neighbor pairing combined with the ZoomOut [Melzi et al. (2019)] technique to estimate the intraoperative displacement field, which is then mapped to the BCs via neural networks. Nikolaev et Cotin (2020) introduced a reduced-order unscented Kalman filter for BC estimation, but their approach is computationally expensive and requires multiple intraoperative samples. These various methods highlight the ongoing challenge of accurately and efficiently estimating boundary conditions from intraoperative data, with each approach offering different trade-offs in terms of robustness and real-time applicability.

7.1.2 Computational Speed

Significant efforts are being directed toward optimizing PBDTs to enhance their computational speed. Various trade-offs between the speed and accuracy of PBDTs have been proposed [Brunet (2020); Haouchine et al. (2013); Niroomandi et al. (2008, 2013)]. Haouchine et al. (2013) suggested using a co-rotational model to handle large deformations with small strain. Yet, their approach significantly loses accuracy when advanced biomechanical laws are considered. Depending on the acceptable level of accuracy loss, reducing the model's degrees of freedom is a viable strategy to meet real-time constraints. Methods

such as Proper Orthogonal Decomposition (POD) [Niroomandi et al. (2008)] and Proper Generalized Decomposition (PGD) [Niroomandi et al. (2013)] have been proposed for this purpose. Another category of methods leverages the high number of cores available in Graphics Processing Units (GPUs) for parallel computing, which enables significant speedups in handling computationally intensive problems [Johnsen et al. (2015)]. Deep neural network architectures have recently demonstrated strong capabilities in learning complex, high-level nonlinear relationships between diverse input-output data [LeCun et Bengio (1998); Scarselli et al. (2009)]. One of the strengths of these networks is their ability to perform inference in real-time when trained with sufficient data accurately. Several works have proposed to train deep neural networks on simulated (using FEM) data [Brunet (2020); Roewer-Despres et al. (2018); Tonutti et al. (2017)], aiming to learn the behavior of PBDTs. U-Mesh, introduced by Brunet (2020), stands out as a simple yet effective solution. U-Mesh is a data-driven approach based on a U-Net [Ronneberger et al. (2015)] architecture, designed to approximate the nonlinear relationship between forces and displacement fields. It is trained in a patient-specific manner using simulated data generated by FEM and achieves real-time performance during inference. U-Mesh [Brunet (2020)] has shown strong performance on real-world data, making it a promising approach in terms of both accuracy and speed. However, U-Mesh is designed assuming that patient-specific characteristics, such as material properties and domain boundary conditions, are known before the intervention. This assumption limits its application in computer-assisted interventions, as these characteristics, when available, are typically only accessible during the interventions. To address this limitation, we propose an extension of U-Mesh that integrates a Hypernetwork [Ha et al. (2017)] to condition U-Mesh [Brunet (2020)] based on the prior distribution of patient characteristics. Therefore, the proposed extension is used upon training in an optimization loop to determine in real-time patient-specific characteristics given intraoperative observations.

7.1.3 Contributions and outlines

In this work, we build on U-Mesh [Brunet et al. (2019)] and introduce a novel approach to enhance the precision of PBDTs by accurately estimating patient-specific characteristics, including material properties and domain boundary conditions. Our contributions are as follows:

- **Flexible Method:** We develop a method adaptable to unknown variables and dynamically adjusts to the underlying physics model, ensuring high precision in various scenarios.

- **Hypernetwork Architecture:** We employ a neural network conditioned on patient-specific characteristics through a hypernetwork architecture, allowing the network to adapt to patient characteristics automatically.

- **Training on Simulation Data:** The neural network is initially trained on simulation data generated using an accurate biomechanical model. This enables it to learn complex relationships between applied forces and relative displacement fields for a given distribution of patient-specific characteristics.

- **Robust and Fast Optimization Process:** After training, given an intraoperative observation, the network identifies the real patient-specific characteristics through a robust and fast gradient-based optimization process that leverages the neural network's learned representations. The architecture incorporates dimensionality reduction for high-dimensional unknown patient-specific characteristics by optimizing over a latent space representation, thereby accelerating the optimization process.

- **Fast and accurate surrogate model** Once estimated, the patient-specific characteristics are injected through the hypernetwork, allowing for a fast and accurate surrogate model of the PBDT.

The chapter is organized as follows: after describing our method in Section 7.2, we illustrate our approach in Section 7.3 by demonstrating its effectiveness across various scenarios involving different physics models, geometries, and patient characteristics. Finally, we discuss our findings in Section 7.5 and conclude in Section 7.6.

7.2 Method

Without loss of generality, we consider the case where a patient-specific PBDT of an organ is needed to assist in surgery. The geometry of the organ can be segmented from a preoperative CT or MRI scan, while the patient-specific properties, denoted by λ, are unknown. These properties, λ, could include material properties and boundary conditions of the geometry. Let f_λ represent the patient-specific PBDT of the organ for a given set of properties λ. For a set of applied forces to the organ, we have denoted F, the relative displacement field is given by $u = f_\lambda(F)$. Our method consists of three steps. First, we build a biomechanical PBDT by employing the FEM, denoted as f_λ^{FEM}. Then, using this biomechanical model, we generate a dataset comprising pairs of forces and displacements, considering a range of distributions for the patient-specific parameters λ. This

dataset is used to build a surrogate model, denoted f_λ^{NN}, which corresponds to an approximation of f_λ^{FEM} with a neural network-based model. We use a hypernetwork to condition the neural network on the patient-specific parameters λ. The network is trained on the dataset generated with the biomechanical model. After training, leveraging the gradient flow of the neural network and its real-time capabilities, we use an optimization process to estimate the patient-specific parameters λ based on given observations, denoted y.

7.2.1 Biomechanical model

The biomechanical model denoted f_λ^{FEM}, is constructed using the same strategy outlined in Section 6.2.2.1 of the previous chapter. This approach involves applying the same principles of continuum mechanics to build the system, which is solved using the FEM. The material properties and boundary conditions will be defined for each of the experiments, see Sections 7.3.1 and 7.3.2.

7.2.2 Surrogate model

This section describes the proposed neural network for learning the PBDT, also called the surrogate model. Our approach involves learning a nonlinear function, denoted f_λ^{NN}, to map the applied forces F to the displacement field u with respect to a set of patient-specific parameters λ. To achieve this, we rely on a hypernetwork architecture. As previously explained in chapter 2, hypernetworks are a class of neural networks that generate the weights of another neural network (usually called target, main, or primary network). They have emerged as a way to enhance the flexibility and performance of deep neural networks [Chauhan et al. (2024)]. Besides their added adaptability, we are primarily interested in the specific gradient flow of such an architecture since both networks are trained in an end-to-end differentiable manner [Ha et al. (2017)]. Training hypernetworks is often challenging, mainly due to the proportionality between input and output magnitudes, which leads to very slow convergence. Ortiz et al. (2023) identified and resolved this issue by treating the predicted weights as additive changes for the primary network. For a given training iteration n, the weights $d\theta_n$ predicted by the hypernetwork h are used to update the weights of the primary network (f_λ^{NN}) following the equation below:

$$\theta_n = \theta_0 + d\theta_n \tag{7.1}$$

With θ_0 the initial weights of the f_λ. Unlike traditional hypernetworks Ha et al. (2017), the weights θ_0 are also trainable parameters. The weights $d\theta_n$ influence the predictions

of f_λ^{NN} by incorporating knowledge of the patient-specific characteristics λ. This strategy permits a better initialization of the primary network weights, which leads to a fast and efficient training of f_λ^{NN}. Typically, the training time of our hypernetwork is comparable to the training time of the primary network on its own.

Figure 7.1 illustrates the proposed neural-network-based method for learning a PBDT conditioned on patient-specific parameters λ. The dimension of the unknown parameters λ can vary: it may be low, such as for stiffness parameters, or very high, such as when determining the Dirichlet boundary condition location on the organ, in which case λ would have a dimension equal to the number of nodes on the organ's mesh. Depending on the dimensionality of λ, we propose two architectures. For low-dimensional λ (leftmost architecture in Figure 7.1), it is directly input into the hypernetwork to generate the additive weights for the primary network f_λ^{NN} (the surrogate model). For high-dimensional λ (rightmost architecture in Figure 7.1), an encoder E is used to reduce the dimensionality to z. The reduced representation z is then input into the hypernetwork h to generate the additive weights and decoded by D to retrieve the patient parameters λ. The role of dimensionality reduction is significant, as it accelerates the optimization process described in section 7.2.3. The following subsections detail the architecture of the neural networks.

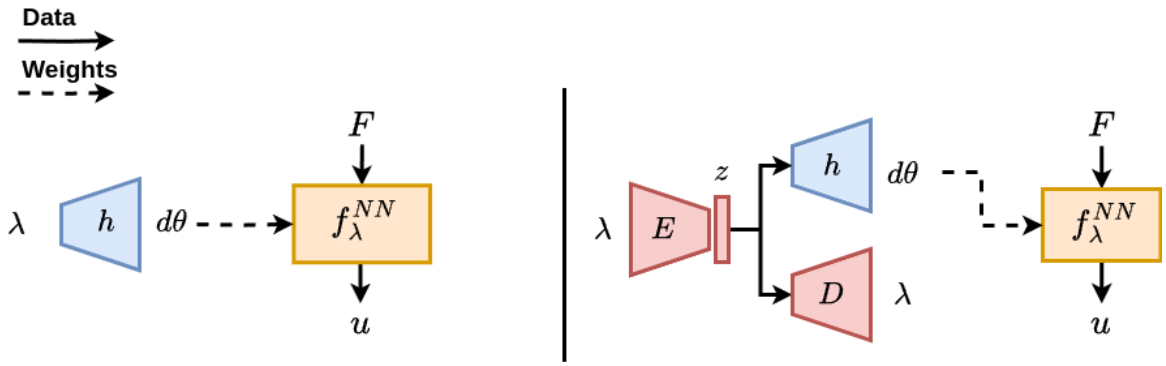


Figure 7.1: Overview of the proposed architecture for learning a PBDT conditioned on patient-specific parameters λ. Depending on the dimensionality of λ, we propose two architectures. **Left**: For low-dimensional λ, it is directly input into the hypernetwork h to generate the additive weights for the primary network f_λ^{NN}. **Right**: For high-dimensional λ, an encoder E is used to reduce the dimensionality. The reduced representation z is then input into the hypernetwork h to generate the additive weights and decoded by D to retrieve the patient parameters λ. Reducing the dimensionality of λ speeds up the optimization process for finding λ. In both architectures, the primary network is U-Mesh [Brunet (2020)], which takes the applied forces as input and predicts the relative displacement field. h, E and D are multi-layer perceptron.

7.2.2.1 Primary network

U-Mesh, a CNN, proposed by Brunet (2020), has shown strong performance in learning Physics-based biomechanical models on real-world data, making it a promising approach in terms of both accuracy and speed. However, U-Mesh is trained on a single value of material properties and Dirichlet boundary conditions. Hence, its accuracy at inference time depends on precise knowledge of patient-specific parameters during training. In this work, by using U-Mesh as a primary network, we provide the knowledge of the patient-specific parameters λ through the hypernetwork h. U-Mesh takes as input the applied forces as a tensor of size $3 \times n_x \times n_y \times n_z$, where n_x, n_y and n_z are the number of nodes in the FEM mesh along the axes x, y and z, respectively. The forces are encoded at each node as a tensor of dimension 3. The network outputs the relative displacement field of the same size. The encoding and decoding paths of the U-Net consist of 3 and 4 layers, respectively. A layer includes a $3 \times 3 \times 3$ convolution filter followed by a $2 \times 2 \times 2$ max pooling operation and ReLU activation function. The number of channels is $[8, 16, 32]$ and $[32, 16, 8, 3]$ for the encoding and decoding paths respectively.

7.2.2.2 Other networks

The hypernetwork h is a Multi-layer perceptron of 4 layers. The last layer is of dimension equal to the number of weights in the U-Mesh, while the first layer's dimension is equal to λ's when λ is for low dimension; otherwise, the input dimension of h is equal to the dimension of λ's latent representation z. A ReLU activation function follows each of h's layers. E and D are also MLPs of four layers each, and they serve as an autoencoder for λ when it has a high dimensionality. A ReLU activation function follows each of their layers. The dimension of the latent space z is experimentally determined based on λ's dimension.

7.2.2.3 Training

The networks f_λ^{NN}, h, E and D are trained end-to-end and supervised using data generated with the physics-based biomechanical model f_λ^{FEM} explained in Sections 7.2.1. We utilize an IBM to discretize the domain, resulting in a regular grid, as shown in the right-hand image of Figure 6.3. This approach is chosen due to its favorable convergence characteristics and regular structure, making it compatible with CNN's architecture chosen for the primary network f_λ^{NN}. Using f_λ^{FEM}, we build a dataset of distinct (F, u, λ) triples, where F represents the applied forces on the organ, u the relative displacement

field, and λ the chosen patient-specific parameters. Since λ corresponds to physical quantities (such as soft tissue stiffness or boundary condition location) in our physics-based problem, we often have statistical data on these parameters from the literature. This knowledge is utilized during dataset generation when available, and otherwise, we choose a range of λ values to encompass its variability, thereby capturing diverse patient characteristics. Section 7.3 presents detailed examples of this process.

The weights of f_λ, E, D and h are optimized by minimizing the loss $\mathscr{L}$ defined in the following Equation:

$$\mathscr{L} = ||u - \hat{u}||_2^2 + \mathscr{L}_\lambda \, . \tag{7.2}$$

In this loss, the first term represents the mean square error between the prediction of the surrogate model $u = f_\lambda^{NN}(F)$ and the ground truth $\hat{u} = f_\lambda^{FEM}(F)$ computed with the biomechanical model. The second term $\mathscr{L}_\lambda$ corresponds to a measure of error between λ and $D(E(\lambda))$ and is included only for the architecture shown in the rightmost image of Figure 7.1, which applies when λ is of high dimensionality.

7.2.3 Patient-specific parameter estimation

Once trained, the surrogate model f_λ^{NN} can predict the displacement field of the organ given the applied forces F and the patient-specific parameters λ. However, λ is usually unknown. In this section, we propose using an optimization process that relies on the trained model f_λ^{NN} to estimate the patient-specific parameters λ. Consider a case where the applied forces are known through an appropriate sensor, typically when using surgical robots like the Da Vinci robot. An observation y of the deformation can also be obtained through an intraoperative (imaging) sensor. For instance, laparoscopic camera, LiDar, ultrasound, IVUS, or CBCT. The observation y may be full or partial imaging of the organ.

Given the observation y of the deformation, we rely on advanced segmentation and reconstruction techniques [El hadramy et al. (2023b); Wasserthal et al. (2023)] to reconstruct the surface of the observation. For instance, point clouds serve as a common representation of any intraoperative observation from the imaging modalities mentioned above. Alongside the applied forces F, the observation y is used to solve an optimization problem to determine the patient-specific parameters λ. This optimization process leverages the trained networks (f_λ^{NN}, h, E and D) and a gradient-based optimizer, such as Stochastic Gradient Descent (SGD), to identify the optimal value λ^* that minimizes an objective function J (described in Equation 7.3 and 7.4), hence maximizing the accuracy of the surrogate model f_λ^{NN} on matching the observation y.

$$\lambda^* = \arg\min_{\lambda} J \tag{7.3}$$

$$J = \mu(g \circ u, y) \quad \text{with} \quad u = f_\lambda^{NN}(F) \tag{7.4}$$

In equation 7.4, g is the preoperative geometry, u the predicted displacement field for a given value of λ and F, y is the point cloud observed intraoperatively, and μ denotes the Euclidean distance between the observed point cloud and the deformed organ. Figure 7.2 illustrates the Euclidean distance μ between an observation y of the ground truth deformed geometry $\tilde{G}$ and the deformation G computed with the surrogate model when λ is incorrectly estimated. For each point in y, its projection on the surface of G is measured to compute the Euclidean distance.

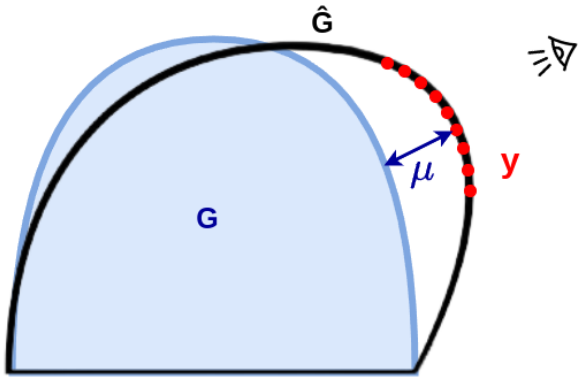


Figure 7.2: Illustration of the distance measure μ described in Equation 7.4. Here, y represents an observation of the deformation $\tilde{G}$, while G denotes the prediction from the surrogate model when λ is misestimated. The objective function is computed as the mean Euclidean distance (μ) between the observed points (y) and the surface of G.

Algorithm 3: Optimization Process

Input : The preoperative geometry G, the applied forces F, the intraoperative observation y, and a prior distribution $\mathscr{P}$

Output: An estimation of λ

$\lambda \sim U(\mathscr{P})$
optimizer $\leftarrow$ SGD(λ, lr)
$J \leftarrow 1 \times 10^9$
while $J >$ *threshold* **do**
    $u_\lambda \leftarrow f_\lambda^{NN}(F)$
    $G \leftarrow g \circ u_\lambda$
    $J \leftarrow \mu(G, y)$
    $\lambda \leftarrow$ optimizer(J)
end

Algorithm 3 shows the optimization process aiming to find the parameters λ that minimizes the objective function in equation 7.4. We begin by setting an initial guess for λ by sampling from a uniform distribution $\mathscr{P}$, the same distribution used for training. During each iteration, the estimation of λ and the applied forces F are used to predict the displacement field u. This displacement field is used to compute G, a deformation of the preoperative geometry g. Therefore, the objective function, defined as the Euclidean distance between G and y, is minimized to estimate the optimal λ. The gradient computation is possible thanks to the chain rule described in equation 7.5 and the automatic differentiation of neural networks. This gradient is the dot product of two terms: (1), which represents the gradient of J with respect to f_λ^{NN} weights, and (2), which corresponds to the gradient of h with respect to its input, note that the first term is identical to the gradient used during training.

$$\frac{\partial J}{\partial \lambda} = \frac{\partial J}{\partial \theta} \cdot \frac{\partial \theta}{\partial \lambda} = \frac{\partial J}{\partial \theta} \cdot \frac{\partial (\theta_0 + d\theta)}{\partial \lambda} = \underbrace{\frac{\partial J}{\partial \theta}}_{(1)} \cdot \underbrace{\frac{\partial (d\theta)}{\partial \lambda}}_{(2)} \tag{7.5}$$

In equation 7.5, the initial weights θ_0 of f_λ^{NN} vanish from the second term as they are independent of the parameters λ. This proposed gradient computation remains robust to any initial guess of λ, as gradient computation is consistently feasible due to the automatic differentiation property of neural networks.

When λ has high dimensionality, we use the architecture on the right-hand side of Figure 7.1. In this scenario, the optimization is performed over the latent-space vector z, which has a lower dimension, thereby reducing the problem's complexity and accelerating the convergence of the optimization algorithm. Upon convergence, the parameter vector $\lambda^* = D(z^*)$ is estimated using the trained decoder D.

7.3 Experiments and results

To evaluate our method, we conducted two experiments to estimate parameters across various scenarios. All experiments involved implementing the neural network and optimization processes using PyTorch[1]. We utilized the Adam optimizer for the neural network training and the optimization process. The FEM simulations were performed using the SOFA Framework [Faure et al. (2012)] with the SOniCS plugin for soft-tissue biomechanics presented in chapter 4. Computation was carried out using an Nvidia Titan RTX GPU. We use the Hausdorff distance in the following experiments to measure the error between the predicted and ground truth meshes. This choice is necessary because the

[1] https://pytorch.org/docs/stable/index.html

predicted and ground truth meshes do not share the same topology. The Hausdorff distance is a suitable metric in this context as it measures the maximum distance between points on one surface and the closest points on the other surface, accurately assessing how well the predicted mesh aligns with the ground truth, even when their structures differ.

7.3.1 Experiment 1: Estimation of soft-tissue characteristics

Accurately identifying the material properties is crucial for developing a precise PBDT model. One of the most challenging aspects is determining the stiffness of the tissue, which is often difficult to acquire and can introduce significant errors in physics-based models. This is due to the inherent variability in biological tissues and the complexity of capturing their mechanical behavior accurately. In this first experiment, we aim to estimate the stiffness of an ex-vivo human liver. By accurately determining the liver's stiffness, we aim to improve the fidelity of the PBDT model, leading to better predictions of tissue deformation under various conditions.

7.3.1.1 Experimental data

An ex-vivo human liver was used for this experiment. Two CT scans were acquired from the liver under two different deformations. As illustrated in Figure 7.3, the first configuration corresponds to a rest state of the organ while in the second configuration a force of 5.1 N was applied on the left lobe. The stiffness of the liver was estimated at 7 kPa using the FibroScan® technique. Additionally, a LiDAR camera captured a partial surface point cloud of the organ in the second configuration.

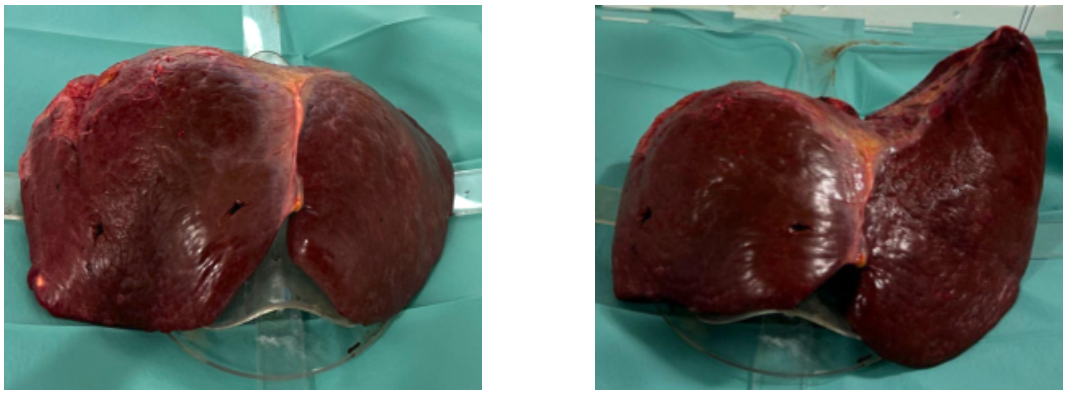


Figure 7.3: Rest and deformed states of the ex-vivo human liver.

7.3.1.2 Biomechanical model (f_λ^{FEM})

We follow the boundary value formulation described in Section 7.2.1 to build f_λ^{FEM}. The geometry was segmented from the CT scan, representing the ex-vivo organ's rest configuration. The liver occupies a volume Ω with boundary Γ. The Dirichlet and Neumann boundary conditions are on Γ_D and Γ_N. Figure 7.4 illustrates the simulation domain. The Dirichlet boundary conditions are assumed to be known and fixed at the interface between the let parenchyma and the portal vein. The material behavior is approximated by using a Saint-Venant Kirchhoff model. See chapter 7.2.1 for the expression of the strain-energy density function.

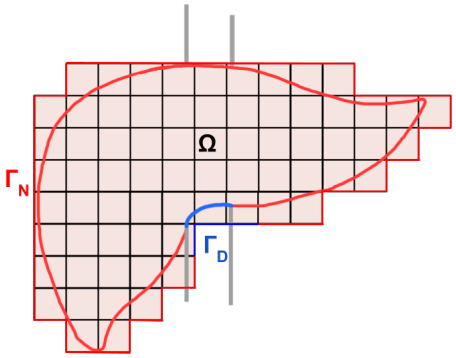


Figure 7.4: The liver occupies a volume Ω with boundary Γ. The Dirichlet and Neumann boundary conditions are on Γ_D and Γ_N. We use an IBM to discretize the domain; this is motivated by its compatibility with U-Mesh's CNN architecture.

7.3.1.3 Data Generation

Using the biomechanical model, denoted f_λ^{FEM}, we generated 3,000 synthetic deformations of the organ. For each deformation, we randomly selected a location on the liver surface to apply traction forces, sampled from a uniform distribution over [2 N, 10 N]. Moreover, Young's modulus was sampled from a uniform distribution over [5000 Pa, 12000 Pa], and the Poisson ratio was set at 0.49 for each generated sample. Consequently, a sample from the dataset includes the applied forces F as input for f_λ^{NN}, the chosen Young's modulus as the input λ of the hypernetwork h, and the relative displacement field obtained through f_λ^{FEM}, which serves as the ground truth to supervise f_λ^{NN}'s prediction u.

7.3.1.4 Network training

In this experiment, the unknown parameter $\lambda = E$ is a scalar representing the material's stiffness. Given the low dimensionality of λ, we use the simpler architecture shown on the

left-hand side of Figure 7.1. The first layer of h is of size 1, h predicts the additive weights $d\theta$ for f_λ^{NN}. These additive weights provide f_λ^{NN} with information about the stiffness. The networks were trained end-to-end on 3,000 generated samples during 400 epochs with a batch size of 1. The loss function is the MSE between the displacement field predicted by the surrogate model (f_λ^{NN}) and the one computed with the biomechanical model f_λ^{FEM} as expressed in Equation 7.2. Note that this experiment has no reconstruction loss $\mathscr{L}_\lambda$ as λ is of dimension 1. Hence, no need for dimensionality reduction.

7.3.1.5 Network results

After training, we evaluated the performance of f_λ^{NN} by applying forces of 5.1 N to the left lobe and comparing the predicted deformation with the ground truth value of the Young's Modulus. We assessed the obtained deformation in terms of Hausdorff distance, and the results are summarized in Table 7.1. Additionally, we reported the Relative Hausdorff distance, which represents the percentage error relative to the deformation amplitude. Our results were compared with those from f_λ^{FEM} and U-Mesh [Brunet (2020)], where U-Mesh was trained with a constant stiffness value. The comparison shows that the surrogate model, f_λ^{NN}, achieves comparable accuracy to both methods while being faster than f_λ^{FEM} and accommodating a range of stiffness values, unlike U-Mesh. Figure 7.5 (left image) displays the deformation predicted by the surrogate model f_λ^{NN} superimposed on an image of the actual deformation. To further assess the robustness of our method, we compared f_λ^{NN} predictions with f_λ^{FEM} solutions across 100 synthetic deformations, yielding a Mean Absolute Error (MAE) of 1.77 mm $\pm$ 0.82 mm. The rightmost image of Figure 7.5 shows a comparison between the predictions of the surrogate (f_λ^{NN}) and biomechanical (f_λ^{FEM}) models for a given applied forces and value of λ. The heatmap represents the error at each location of the geometry.

7.3.1.6 Optimization results

In this section, we evaluate the optimization method for estimating the stiffness of the ex-vivo liver. Given the applied forces at the left lobe, a point cloud of the liver surface was captured using LiDAR. This point cloud is the observation y in the optimization method described in Section 7.2.3. The Adam optimizer efficiently converged after 10 iterations, with an average computation time of 46 ms per iteration. The estimated stiffness value obtained was $\lambda = 6.8$ kPa, leading to an error within 3% of the ground truth value (equals 7 kPa). This result demonstrates the effectiveness of our optimization approach in accurately estimating material properties. The Adam optimizer's efficiency and the esti-

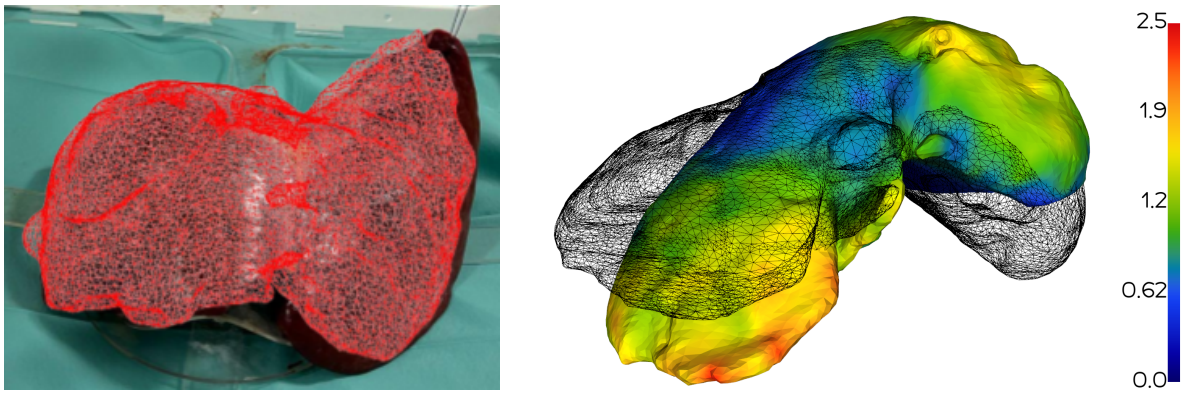


Figure 7.5: Left: In the wireframe visualization, the prediction of f_λ^{NN} is shown when applying to the rest state (left image of Figure 7.3) the same loads that produced the deformed state (right image of Figure 7.3). The predicted result is superimposed over the ground truth deformation image for comparison. **Right**: The wireframe illustrates the liver's rest shape. The surface mesh represents a f_λ^{NN} prediction when specific loads are applied to this rest shape. The heatmap shows the errors of f_λ^{NN} compared to the f_λ^{FEM}'s solution under the same loads.

	Hausdorff (mm)	**Relative Hausdorff (%)**	**Time (ms)**
f_λ^{FEM}	**15.1**	**11.6**	**500**
U-Mesh [Brunet (2020)]	**16.6**	**12.7**	**4**
f_λ^{NN}	**16.5**	**12.6**	**4**

Table 7.1: Results of the liver experiments compare the predictions of standard FEM, U-Mesh, and f_λ^{NN} against the ground truth deformation. f_λ^{NN} achieves results comparable to both state-of-the-art methods while being significantly times faster than standard FEM and more versatile than U-Mesh [Brunet (2020)] in handling a range of material properties.

mated value's accuracy underscore our method's robustness in handling real-world data and providing reliable parameter estimation.

7.3.2 Experiment 2: Boundary condition estimation for soft tissue biomechanics

The BCs play a crucial role in the PBDT accuracy as they define the interactions between the organ and its environment. The problem of estimating zero-displacement boundary conditions is challenging since it requires identifying which nodes of the finite element model are constrained. Consequently, the vector λ, over which the optimization takes place, is very large and can contain tens of thousands of unknowns depending on the

mesh's size. In such a case, many optimization methods would fail or be time-consuming. In this section, we propose to estimate the zero-displacement boundary conditions using the proposed method.

7.3.2.1 Experimental data

: For this experiment, we use data from Mazier et al. (2022). The dataset consists of a cylinder PolyDiMethylSiloxane (PDMS) beam that deforms under gravity with its left extremity fixed to a vertical support. The non-deformed geometry is provided, and the material behavior is approximated by a nearly incompressible Mooney-Rivlin [Rivlin (1948)] model with parameters C_{01} = 101 kPa and C_{10} = 151 kPa. This approximation was performed by Mazier et al. (2022) using the Mach-1™ mechanical testing system (Biomomentum, Canada). Additionally, an image of the deformation was captured, as illustrated in the left-hand image of Figure 7.8.

7.3.2.2 Mechanical model (f_λ^{FEM})

We follow the boundary value formulation in 7.2.1 to build the biomechanical PBDT, denoted f_λ^{FEM}, for the PDMS beam. The beam occupies a volume Ω with boundary Γ. The Dirichlet and Neumann boundary conditions are on Γ_D and Γ_N, two subsets of Γ. Γ_D is the unknown to be determined. Figure 7.6 illustrates these domains.

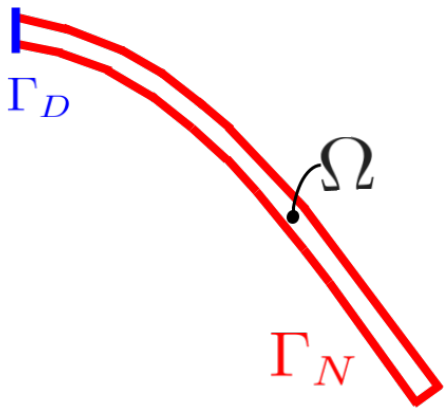


Figure 7.6: The beam occupies a volume Ω with boundary Γ. The Dirichlet and Neumann boundary conditions are on Γ_D and Γ_N.

The material behavior is approximated with a nearly incompressible Mooney-Rivlin [Rivlin (1948)], see chapter 3 for the expression of the strain-energy density function W.

7.3.2.3 Data generation

We generated 6,000 training samples using the finite element simulation described earlier. For each sample, body forces, denoted F, were selected from a uniform distribution over [15 N, 7 N], and a section of the cylinder was randomly chosen to represent Γ_D. The relative displacement field is $u = f_\lambda^{FEM}(F)$, with λ a discrete representation of Γ_D over the beam's mesh. This representation is a vector where the dimension equals the number of nodes in the FEM mesh (1088 in our case). Each element of the vector is set to 1 if the corresponding node is in the Γ_D domain, and 0 otherwise. We illustrate an example of generated deformations for different chosen locations of Γ_D in Figure 7.7.

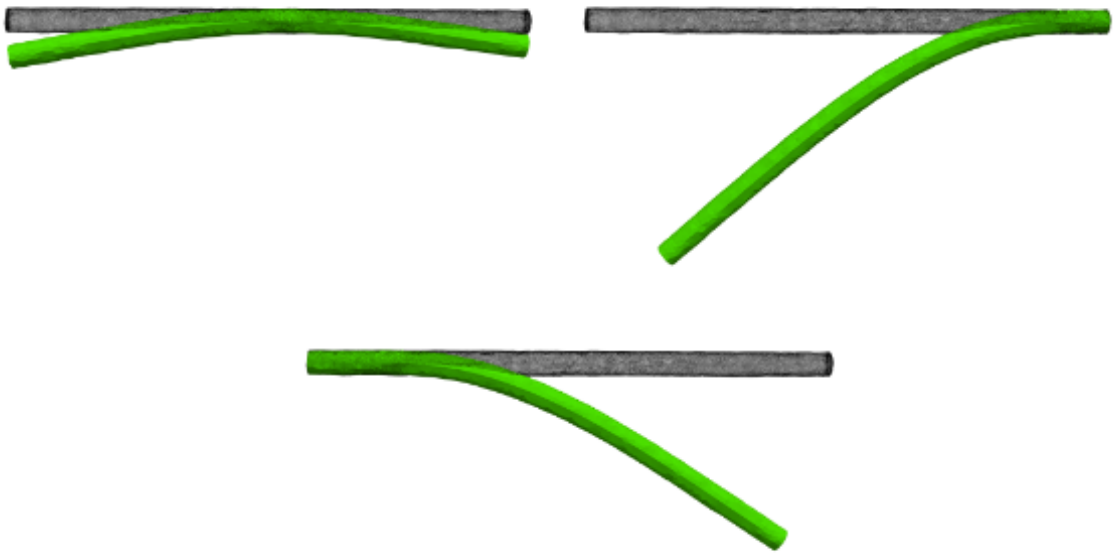


Figure 7.7: Illustration of three examples of deformations generated for training. The Dirichlet section is set in the middle, left, and right, respectively.

7.3.2.4 Network training

In the case of this experiment, the parameter λ is a discrete representation of the Dirichlet boundary condition over the beam's mesh, resulting in a high dimensionality of 1088.

	Hausdorff (mm)	Relative Hausdorff (%)	Time (ms)
f_λ^{FEM}	**5.8**	**4.4**	**3000**
U-Mesh [Brunet (2020)]	**6.2**	**4.7**	**4**
f_λ^{NN}	**6.4**	**4.9**	**4**

Table 7.2: Results from the beam experiments, shows comparisons between predictions made by the biomechanical model (f_λ^{FEM}), U-Mesh, and the surrogate model (f_λ^{NN}) against the ground truth deformation. f_λ^{NN} achieves results comparable to both state-of-the-art methods while operating 750 times faster than the biomechanical model and offering greater flexibility than U-Mesh [Brunet (2020)] in handling diverse patient characteristics.

Therefore, we use the architecture shown on the right-hand side of Figure 7.1, which includes a dimensionality reduction of λ. The input size of E and the output size of D equal 1088, the latent space representation z having a dimension of 3 (Experimentally chosen). The networks were trained end-to-end during 400 epochs on the 6,000 generated samples, with a batch size of 1. We used a Binary Cross Entropy for the reconstruction loss of λ, denoted as $\mathscr{L}_\lambda$ in equation 7.2.

7.3.2.5 Network results

Aiming for comparison with real data, upon training, we have evaluated the surrogate model (f_λ^{NN}) when the body forces are the gravity and the Dirichlet domain is on the left extremity of the beam. The resulting deformation was compared with the real from Mazier et al. (2022) and is illustrated in the right-hand image of Figure 7.8. Similar to the previous experience, we use Hausdorff metrics to compare the predictions and the ground truth geometries. Table 7.2 presents the numerical results of f_λ^{NN}, we also report the performances of the biomechanical model f_λ^{FEM} and U-Mesh [Brunet (2020)] both compared with the ground truth deformation from Mazier et al. (2022). Here, U-Mesh was trained on constant Dirichlet boundary conditions. Results show that f_λ^{NN} achieves comparable results with both f_λ^{FEM} and U-mesh, while operative 750 times faster than f_λ^{FEM} and being more versatile than U-Mesh in terms of Dirichlet boundary condition location.

To further assess the robustness of f_λ^{NN}, we have compared its results with f_λ^{FEM} over 100 deformations with various locations of Dirichlet boundary conditions. Results achieved a MAE of $0.67 \pm 0.57mm$ with a maximum and minimum MAE of $2.7mm$ and

$0.12mm$, respectively. We illustrate in Fig. 7.9 three results, where we compare the prediction of f_λ^{FEM} and f_λ^{NN} on cases where the Dirichlet boundary conditions are fixed on the left, middle, and right cross sections respectively.

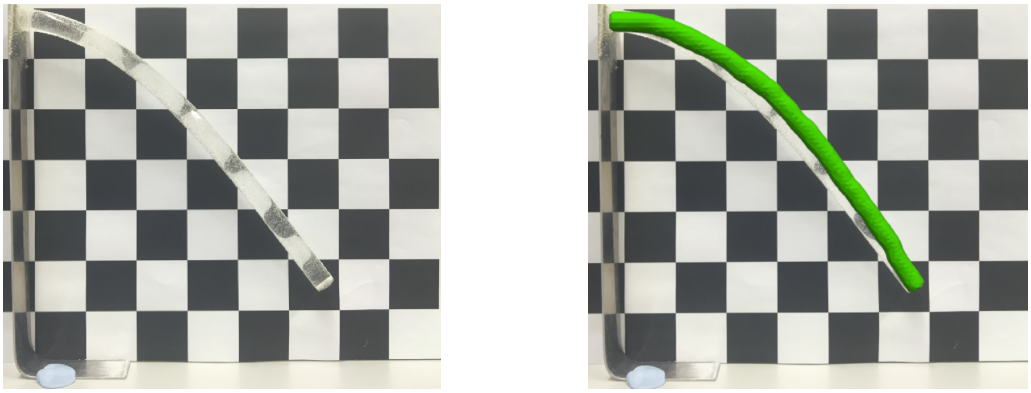


Figure 7.8: Left: the observed deformation of the beam, fixed on the left side and deforming under gravity. **Right**: prediction of the neural network f_λ (in green) overlaid onto the ground truth beam.

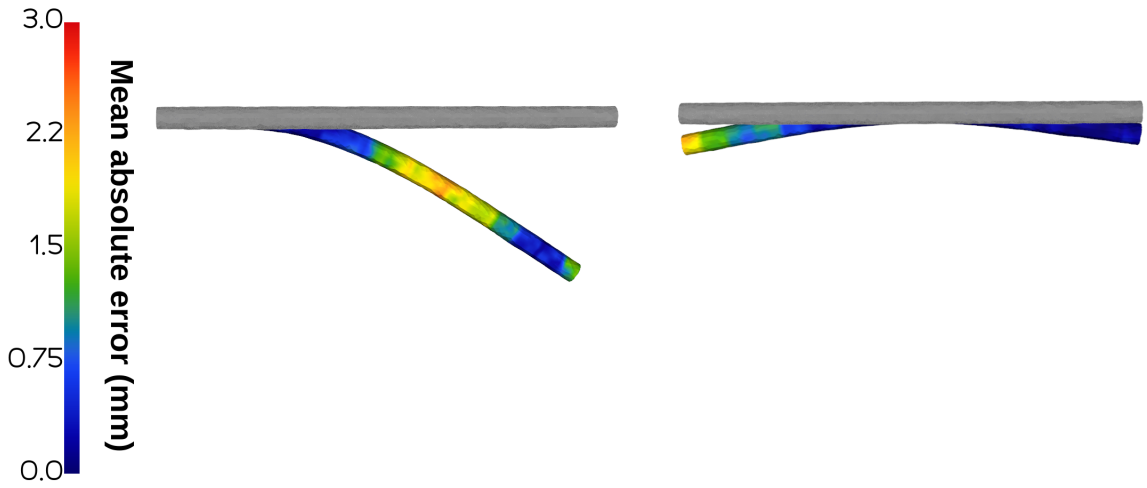


Figure 7.9: Comparison of the surrogate (f_λ^{NN}) and biomechanical (f_λ^{FEM}) models under various deformations and different Dirichlet boundary conditions: In gray, the non-deformed shape of the geometry, the errors between f_λ^{FEM} and f_λ^{NN} are displayed on the deformed beam. Two examples are illustrated. **Left**: The Dirichlet boundary conditions are on the left extremity. **Right**: The Dirichlet boundary conditions are on a cross-section in the middle of the geometry.

7.3.2.6 Optimization results

An important contribution of this work is the ability to leverage the surrogate model (f_λ^{NN}) to determine the parameters λ that best fit an observation y. In this experiment, we use

the image of the deformation shown in Figure 7.8 (left image) as an observation. Mazier et al. (2022) reconstructed and scaled the deformed geometry from this image. We used a point cloud of the visible surface as an observation y for the optimization process described in Section 7.2.3. This point cloud is a constraint for the deformation predicted by f_λ^{NN} during the optimization process. The optimization over z is performed using the Adam optimizer with a learning rate of 2.9 and a random initial guess of z. Convergence was achieved after 10 iterations, yielding estimations z^* decoded using D to estimate λ. This estimation was input into E and used to predict the deformation of the beam under gravity. Comparison with the ground truth resulted in a Hausdorff of 5.8 mm. The optimization over the trained neural network takes, on average, 49 ms per iteration (40 ms for the forward step and 9 ms for the backward step)

7.4 Ablation study

7.4.1 Dimensionality reduction

In cases where the patient characteristics λ have a high dimension, we use an autoencoder (comprising encoder E and decoder D) to reduce the dimension to a latent representation z. This approach was applied in Experiment 1. The optimization is then performed over the latent representation z, accelerating the optimization algorithm's convergence. To justify this, we ran the optimization process in Experiment 2 over the native vector λ without dimensionality reduction, where λ had a dimension of 1088. In this scenario, the optimization algorithm converged after 150 iterations, compared to just 10 iterations when dimensionality reduction was used. Each iteration takes 49 ms. Thus, the total time for estimating the boundary conditions without dimensionality reduction is 7350 ms, whereas it is only 490ms when the optimization is performed over the latent representation.

7.4.2 HyperNetwork architecture

The proposed architecture uses a special type of Hypernetworks, as explained in Section 7.2.2. Unlike traditional Hypernetworks [Ha et al. (2017)], where the predicted weights completely replace the primary network's weights, our approach uses the predicted weights as additive adjustments to the primary network's weights. This strategy enables faster and more stable training. To support this, we conducted Experiment 1 using both approaches and reported the training losses in Figure 7.10. The Additive Weights Hypernetworks training has a higher convergence speed and is more stable.

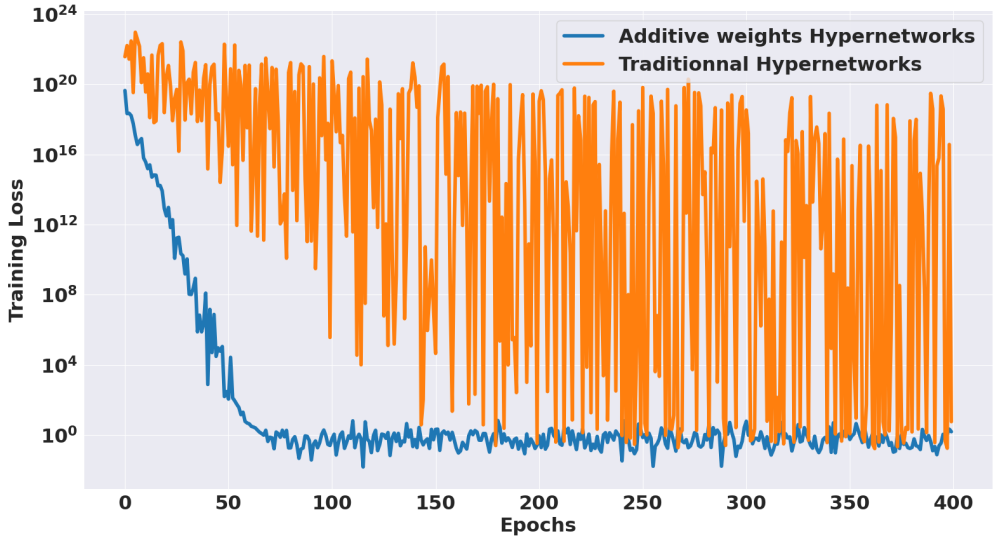


Figure 7.10: Comparison of training losses using two hypernetwork strategies. In orange, the traditional hypernetwork strategy is employed, where the primary network's weights are fully predicted by the hypernetwork. In blue, the hypernetwork predicts additive changes to the primary network's weights. In this work, we adopt the second strategy, as it enables faster and more stable training.

7.5 Discussion

Results show that the proposed method is effective in scenarios involving complex material properties and anatomical structures. The experiments demonstrated that the surrogate model f_λ^{NN} achieves comparable accuracy to both biomechanical model f_λ^{FEM} and U-Mesh while operating significantly faster than the biomechanical model f_λ^{FEM}. Moreover, unlike U-Mesh, which assumes knowledge of patient characteristics before training, f_λ^{NN} demonstrates versatility over a distribution of patient characteristics. This flexibility allows f_λ^{NN} to adapt to patient-specific characteristics intraoperatively, enhancing its robustness and reliability in clinical applications where such variations are prevalent.

The optimization process was effective across both scenarios tested, showcasing its ability to identify patient-specific characteristics that improve the surrogate model's predictive accuracy. Since our primary goal is to ensure the surrogate model's precision, we validated the optimization process based on its ability to find characteristics that yield accurate predictions when used in the surrogate model. This validation confirmed that the proposed approach maintains high accuracy and offers a robust and efficient solution for real-time, patient-specific modeling in clinical settings. Moreover, we utilized point cloud data as intraoperative observations. However, in some scenarios, point clouds may not

fully capture the deformation of a 3D geometry, as they only describe the surface of the geometry. We could explore using 3D data from advanced imaging modalities like ultrasound for more comprehensive representation. These in-depth modalities could provide more detailed insights into the internal deformations and characteristics, potentially enhancing the accuracy of the optimization process and, hence, providing more accurate patient characteristics for the surrogate model.

It is worth noting that our architecture is not limited to a specific choice of primary network. While we have chosen U-Mesh for its efficiency, we have also explored other architectures, including Physics-Informed Neural Networks (PINNs) [Raissi et al. (2019)], and found them to integrate well as a primary network within the proposed framework. This demonstrates that our method's flexibility extends beyond a specific choice of the primary network, making it adaptable to various neural network architectures suited for different clinical applications.

7.6 Conclusion

In this chapter, we present a novel method to improve the accuracy of physics-based digital twins (PBDTs) by precisely estimating patient-specific characteristics based on intraoperative observations. Our approach utilizes a hypernetwork architecture to condition neural network-based surrogate models on patient-specific properties, including material characteristics and boundary conditions. Initially, the network is trained using simulation data from a biomechanical model, accommodating a range of patient characteristics. After training, we employ a gradient-based optimization process to refine the surrogate model, thereby determining the optimal patient-specific parameters that align with the given intraoperative observation. The proposed optimization algorithm converges in real-time, a crucial feature for effective integration into clinical settings.

CHAPTER 8

CONCLUSION

8.1 Contributions

This thesis develops a novel IVUS-guided augmented reality (AR) system for minimally invasive hepatic procedures. In fact, despite the recognized advantages of intravascular ultrasound (IVUS) in providing intraoperative imaging, to our knowledge, there has been no method to integrate IVUS data into surgical guidance systems for liver surgery until now. The system developed in this thesis aims to enhance surgical precision by integrating real-time IVUS data with preoperative imaging, offering surgeons an augmented view of the liver's internal structures. This can be achieved by registering IVUS data with preoperative CT scans, providing an up-to-date and anatomically accurate visualization during surgery, which adapts to organ deformation.

In Chapter 1, we presented a comprehensive review of registration methods, focusing on intensity- and feature-based techniques. While intensity-based registration is often used in medical imaging, it poses challenges in multi-modality scenarios, such as registering IVUS data with CT. To overcome this, we employed feature-based registration methods in this thesis. In conjunction with biomechanical models, feature-based approaches allow us to address the registration problem by circumventing multi-modality issues and taking advantage of the biomechanical model's ability to extrapolate over sparse and noisy intraoperative data, like that from IVUS. This decision forms the backbone of our registration process.

Throughout this work, several key contributions have been made to advance the use of IVUS-based augmented reality in surgery. Early chapters provided the theoretical groundwork necessary to understand the technologies employed in this thesis. Chapter 2 explored the fundamentals of deep learning, beginning with basic concepts like perceptrons and multilayer perceptrons (MLPs), and progressing to advanced architectures such as Convolutional Neural Networks (CNNs), Long Short-Term Memory networks (LSTMs), and Hypernetworks. These models are central to the methods developed in this thesis, enabling the analysis and interpretation of IVUS data and its integration with preoperative imaging. Chapter 3 introduced the essential principles of soft-tissue computational biomechanics, focusing on the finite element method (FEM) as a tool for simulating complex biomechanical problems. This knowledge allowed for the construction of a detailed biomechanical liver model that plays a crucial role in the non-rigid registration process between preoperative CT data and intraoperative IVUS imaging in the context of this thesis.

One of the core technical contributions is the development of SOniCS, a framework introduced in Chapter 4 that simplifies the implementation of hyperelastic material mod-

els for simulating soft tissues. By integrating FEniCS, a tool for solving partial differential equations, with SOFA, a framework for real-time simulation of deformable objects, SOniCS enables efficient and accurate modeling of complex tissue behavior. This biomechanical model, derived from preoperative CT data, provides the necessary framework for registering the intraoperative IVUS data, ensuring accurate alignment and adaptability to intraoperative conditions. In Chapter 5, we tackled the challenge of IVUS volume reconstruction without relying on external tracking systems. Traditional tracking systems are limited in minimally invasive surgeries due to size constraints. In addition, they can be expensive and prone to interference from surgical instruments. To overcome these limitations, we developed a novel AI-driven method for reconstructing 3D IVUS volumes, eliminating the need for electromagnetic tracking systems. This approach simplifies the surgical setup and reduces the costs of the intraoperative imaging system, making it more practical for real-world use. Chapter 6 presented the thesis's primary contribution: a method for real-time registration of intraoperative IVUS volumes with preoperative CT data. This method leverages deep learning, specifically a neural network trained on data generated by the biomechanical liver model, to align intraoperative data with the preoperative model. This enables dynamic adjustments to the preoperative models during surgery. In Chapter 9, we extended this registration framework further by adapting it to account for patient-specific variations. Recognizing that biomechanical models often rely on assumptions about material properties and boundary conditions that can vary between patients, we enhanced the neural network to predict these variables in real-time. This process incorporates an optimization loop that identifies the material properties and boundary conditions that best match the intraoperative observations. This allows the biomechanical model to adapt to each patient's unique mechanical characteristics dynamically.

In the appendix A, we present a work that, while not directly related to the primary subject of this thesis, was developed during the early stages of the PhD program when IVUS data was not yet available. This work aimed to improve CT guidance during needle-based liver procedures, such as tumor ablation while reducing the need for contrast agent injections during these interventions. The approach involved augmenting intraoperative CT images with a preoperative vascular network, deformed to match the current acquisition.

8.2 Limitations and future work

While this thesis has developed the core components of an IVUS-guided AR system for liver surgery, certain limitations must be acknowledged. The primary limitation is the fact that the full pipeline has not yet been validated with real IVUS data. In Chapter 6, the validation was performed using fully synthetic data generated from preoperative CT scans by introducing deformations, partiality, and noise to mimic intraoperative conditions. This synthetic data was important for testing the developed methods in the absence of real IVUS datasets but did not fully replicate the complexities of real intraoperative imaging.

Despite this, our results on synthetic data showed promise. In Chapter 6, we compared our method against the U-Mesh framework proposed by Brunet et al. (2019), which has been validated on real clinical data. Our approach achieved better results in terms of accuracy on synthetic datasets, demonstrating its potential effectiveness. However, it is important to note that U-Mesh's validation on real data adds significant credibility to its performance. The fact that our method outperforms U-Mesh in synthetic tests suggests that it is highly promising for real-world applications. However, this hypothesis remains to be confirmed with real IVUS data.

To validate the proposed method, a specific protocol for data acquisition is required. First, a preoperative CT scan of the liver under an initial deformation must be obtained. Intraoperatively, an IVUS sweep is needed to capture a sequence of ultrasound images corresponding to the liver's intraoperative deformation. Additionally, an intraoperative CT scan should be acquired to serve as the ground truth for validation. After applying the registration between the preoperative CT and the intraoperative IVUS images, the resulting deformation of the preoperative CT will be compared to the intraoperative CT. Therefore, it is crucial that both the intraoperative IVUS and the intraoperative CT are captured under the same deformation of the organ to ensure an accurate comparison.

This acquisition protocol presents several challenges. First, the IVUS probe is not commonly used in liver surgery, as it is not a standard tool in real human interventions for this type of procedure. As a result, its integration into clinical workflows is still in development. When considering animal models, additional difficulties arise, such as the challenge of acquiring an intraoperative IVUS sequence and an intraoperative ground truth CT under the same deformations. Animal respiration and movement between the two acquisitions often result in variations in the liver's deformation, making it difficult to capture consistent and reliable data for validation purposes. These factors make the acquisition protocol complex, but essential for validating the effectiveness of the method.

Moving forward, the next step is clear: to validate the full pipeline with real intraop-

erative IVUS data. This will allow for a more comprehensive evaluation of our system's performance under true clinical conditions, where factors such as noise, tissue variability, and the presence of surgical instruments can introduce new challenges. Acquiring real IVUS data for liver surgeries and validating the system in a clinical setting will be crucial to demonstrating its robustness for practical use. Once the full pipeline is validated on real IVUS data, an important direction for future work will focus on refining the augmented reality (AR) display system. After successfully registering the preoperative model with the intraoperative IVUS volume, the next step involves accurately overlaying the deformed preoperative model onto the laparoscopic view. This AR overlay is crucial for providing real-time, visual guidance to surgeons, helping them precisely track tumors and navigate critical anatomical structures during surgery. However, this additional registration step between the laparoscopic view and the deformed preoperative model is expected to be rigid. Since the preoperative model undergoes deformation through its alignment with the IVUS data, no further non-rigid adjustments to the model are necessary at this stage. The key challenge is ensuring that the deformed model, which already accounts for anatomical changes, is rigidly aligned with the laparoscopic camera's field of view. By achieving this, the augmented reality system can display a synchronized, real-time view that accurately reflects the patient's anatomy.

If the available laparoscopic images contain depth information, this rigid registration can be performed by registering a point cloud computed from the laparoscopic view with the 3D deformed preoperative model. A rigid registration method like Iterative Closest Point (ICP) can align the two datasets. In cases where depth information is not directly available, alternative methods exist to extract a point cloud from monocular images. One such method is Simultaneous Localization and Mapping (SLAM) [Campos et al. (2021)], which uses the motion of a single camera to map the environment and generate a point cloud. Another approach is to leverage deep neural networks, such as MiDaS [Ranftl et al. (2022)], which compute depth directly from monocular images. SLAM and MiDaS provide feasible ways to generate point clouds from standard 2D laparoscopic images, enabling the required data to solve a rigid registration problem, even without dedicated depth-sensing equipment.

While the natural future work of this thesis is the validation of the full pipeline using real intraoperative IVUS data, other directions for improvement can also be pursued by focusing on individual components of the pipeline. For instance, the developed registration method in this thesis is patient-specific, relying on a neural network trained on data generated from a patient-specific biomechanical model built using preoperative data. As a result, the network is tailored to work only for the specific geometry of the patient. Future work could address this limitation by incorporating neural network architectures that

can be trained on a variety of patient anatomies, making the system more robust to differences in geometry across patients. One potential approach could involve using graph neural networks or transformers, which are well-suited to handle variations in geometries and topologies, allowing for training across multiple liver anatomies. Another promising idea would be to condition the current neural network with the patient's geometry through a hypernetwork. In this approach, the hypernetwork would take a representation of the patient's geometry as input and generate the weights that enable the primary network to register intraoperative data for that specific patient. This method would be similar to the hypernetwork proposed in Chapter 9, where it was used to adapt to material properties and boundary conditions. Incorporating these advanced techniques could significantly enhance the versatility and applicability of the registration system across a wider range of patients.

Finally, In Chapter 9, we utilized a hypernetwork to condition the primary neural network on either the material properties or the boundary conditions of the biomechanical liver model. This approach allows the network to adapt to variations in one patient-specific characteristic, ensuring it can effectively register intraoperative data within that specific context. However, this means the network is trained to handle only one type of variation at a time. In future work, we could extend this approach by conditioning the network on both material properties and boundary conditions simultaneously. To achieve this, we could introduce a heterogeneous parameter λ, representing multiple patient-specific quantities, such as tissue stiffness and organ boundary conditions. By conditioning the network on λ, we could create a more robust model capable of handling complex, multi-factor variations, thus improving the generalizability of the method across different patient anatomies.

CHAPTER 9

A BRIEF SUMMARY IN FRENCH

9.1 Introduction

La chirurgie du foie est une procédure complexe et délicate en raison de l'anatomie complexe de cet organe et de sa proximité avec des structures vasculaires critiques. Les interventions hépatiques nécessitent une précision extrême pour éviter les saignements, minimiser les dommages aux tissus sains et assurer une résection efficace. De plus, le foie est un organe mou et mobile, se déplaçant avec la respiration, ce qui complique encore la planification et la navigation chirurgicale en temps réel. L'identification précise des structures internes, telles que les veines et les artères hépatiques, est cruciale mais difficile à réaliser avec les techniques d'imagerie conventionnelles seules, surtout en laparoscopie où la visibilité est réduite.

L'intégration de la réalité augmentée (RA) dans la chirurgie hépatique pourrait apporter des avantages significatifs pour surmonter ces défis. La RA permet de superposer des images préopératoires en temps réel, comme des scans CT, directement sur la vue chirurgicale. En fournissant un guide visuel amélioré, cette technologie aide les chirurgiens à localiser précisément les structures critiques et les lésions tumorales, tout en réduisant la dépendance aux imageries intermittentes. La RA permettrait d'améliorer la précision des gestes, et de minimiser les risques de complications, contribuant ainsi à la sécurité et à l'efficacité globales de l'intervention.

Cependant, le foie se déforme continuellement pendant l'intervention en raison de la respiration, de l'interaction avec les instruments chirurgicaux, ainsi que d'autres facteurs liés à la manipulation et à la physiologie du patient. Par conséquent, l'imagerie CT préopératoire, bien qu'initialement précise, ne reflète plus ces déformations dynamiques du foie une fois l'intervention commencée. Cette discordance entre l'image préopératoire et la réalité anatomique en temps réel compromet la précision de la RA au cours de la chirurgie.

Pour maintenir une RA fiable et précise tout au long de l'intervention, il est essentiel de combiner les données CT préopératoires avec une modalité d'imagerie intraopératoire en temps réel. Cette fusion se réalise grâce à un processus de recalage entre les données préopératoires et intraopératoires, permettant d'aligner et de mettre à jour le CT avec la déformation actuelle du foie en continu. Ce recalage en temps réel fournit aux chirurgiens une vue augmentée toujours fidèle à l'état anatomique du patient, et leur permet ainsi de naviguer avec une plus grande précision et de réduire les risques d'erreur tout au long de l'intervention.

Parmi les modalités d'imagerie intraopératoire et en temps réel, on trouve la caméra laparoscopique et l'échographie. La caméra laparoscopique offre une vue en surface des

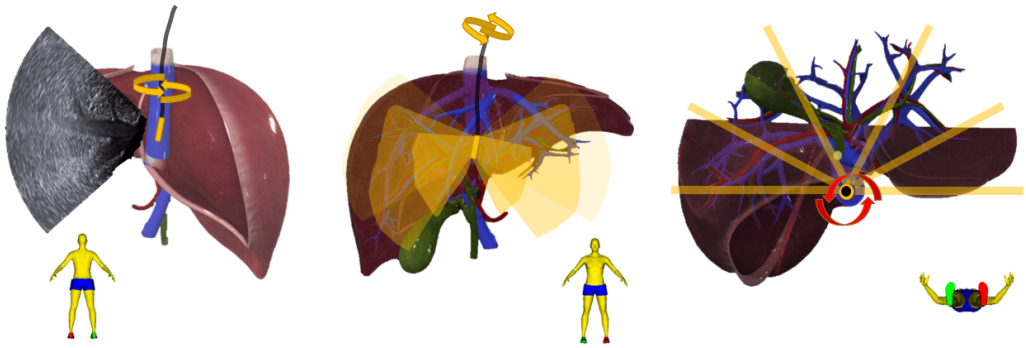


Figure 9.1: Gauche : cathéter IVUS positionné dans le lumen de la veine cave inférieure à la surface postérieure de l'organe, avec un exemple d'images obtenues par émission latérale et formation de faisceau longitudinal; **Centre** : vue antérieure du foie montrant les mouvements de rotation du cathéter permettant des images en coupe complète; **Droite** : vue inférieure illustrant les acquisitions ultrasonores par rotation.

organes, mais elle ne permet pas de visualiser les structures internes du foie, ce qui limite son utilité pour une navigation précise à l'intérieur de l'organe. L'échographie, en revanche, se révèle plus adaptée, car elle permet une exploration interne du foie et une meilleure visualisation des structures vasculaires et tumorales. L'échographie laparoscopique (LUS) est souvent utilisée, mais elle présente des limitations importantes dans le cadre d'interventions mini-invasives : elle occupe en continu un des trocarts nécessaires à l'intervention, mobilise une main du chirurgien, et sa position en contact direct avec la surface du foie entraîne une déformation de l'organe. Cette déformation peut comprimer les vaisseaux et perturber l'interprétation des images, rendant l'utilisation de la LUS moins optimale pour une assistance chirurgicale en temps réel dans des conditions mini-invasives.

Bien que l'intravascular ultrasound (IVUS) soit principalement utilisé dans les interventions cardiovasculaires, cette technologie pourrait également offrir des avantages significatifs dans le cadre de la chirurgie hépatique. L'IVUS se compose d'une sonde ultrasonore montée à l'extrémité d'un cathéter, permettant de naviguer à l'intérieur du lumen de la veine cave inférieure (VCI) (voir Figure 9.1). En positionnant le cathéter dans la VCI et en le dirigeant vers les principales structures vasculaires, l'IVUS permet une visualisation interne des vaisseaux hépatiques et de l'architecture interne du foie, sans nécessiter un accès direct ou une pression sur la surface de l'organe. Urade et al. (2021) ont démontré la faisabilité de cette approche lors d'expériences menées sur des cochons, montrant que l'IVUS permet de naviguer et d'observer l'ensemble des structures internes du foie. Cette technique pourrait ainsi fournir aux chirurgiens une solution pour obtenir une imagerie continue et non déformante des structures hépatiques en temps réel, ce qui améliorerait la précision et la sécurité des interventions mini-invasives sur le foie.

Cette thèse comble un manque en développant un système de RA guidé par IVUS pour la chirurgie mini-invasive du foie. L'objectif est d'améliorer la précision chirurgicale en intégrant l'imagerie IVUS en temps réel avec la technologie de RA, superposant les données d'imagerie préopératoire directement sur le champ opératoire. En exploitant les capacités d'imagerie synchrones et continues de l'IVUS, qui fournissent une vue complète et en profondeur du foie en rotation du cathéter, ce système offrirait une perspective détaillée des structures internes de l'organe, y compris l'arbre vasculaire et les tumeurs. Une telle RA est rendue possible par le recalage des données IVUS intraopératoires avec des images médicales préopératoires comme le scanner ou l'IRM. Ce processus de recalage permet des ajustements dynamiques des modèles préopératoires, garantissant qu'ils reflètent avec précision l'état anatomique actuel du foie, et répond ainsi aux défis tels que la déformation de l'organe et les limitations de visualisation. Pour aborder le problème de recalage, nous utilisons un recalage basé sur des caractéristiques afin de contourner les problèmes liés à la multi-modalité et au choix des métriques de similarité. En outre, nous employons un recalage basé sur un modèle en tirant parti d'un modèle biomécanique du foie construit à partir des données préopératoires, un choix motivé par la capacité de ces modèles à extrapoler efficacement à partir de données intraopératoires limitées et bruitées, comme celles obtenues par IVUS.

Cette thèse est structurée comme suit : le chapitre 2 couvre les bases de l'apprentissage profond et présente les architectures essentielles pour la compréhension des concepts développés dans cette thèse. Chapitre 3 présente les fondements de la mécanique des milieux continus appliquée aux tissus mous, en expliquant notamment la relation entre contraintes et déformations, ainsi que la formulation de systèmes mécaniques résolus par la méthode des éléments finis (FEM). Ensuite, chapitre 4 introduit SOniCS, un cadre permettant d'intégrer FEniCS, dédié à la résolution d'équations différentielles, avec SOFA, utilisé pour la simulation en temps réel d'objets déformables, ce qui facilite le développement de modèles hyperélastiques. Le chapitre suivant propose une méthode de reconstruction volumique basée sur l'intelligence artificielle, permettant de reconstruire un volume IVUS 3D à partir d'une séquence d'images sans nécessiter de suivi. La contribution principale de cette thèse est détaillée dans le chapitre 6, avec une nouvelle méthode de recalage permettant d'aligner les données CT préopératoires aux données IVUS intraopératoires en utilisant un modèle biomécanique construit avec SOniCS. Chapitre 9 introduit une méthode permettant d'augmenter la précision de ce modèle en identifiant des propriétés spécifiques au patient pour améliorer l'alignement des données. Enfin, la thèse se conclut par une discussion sur les limites des méthodes développées et des perspectives pour les travaux futurs. En annexe, un travail parallèle mené au début de cette thèse est présenté; bien que non directement lié au sujet principal, ce projet a con-

tribué à une meilleure compréhension des techniques de recalage et a donné lieu à une publication à MICCAI 2023.

9.2 SOniCS

Dans le cadre de cette thèse, nous avons développé le plugin SOniCS (SOFA + FEniCS) pour créer un modèle biomécanique du foie qui puisse répondre aux exigences de précision nécessaires pour le recalage des données intra-opératoires d'IVUS avec les données préopératoires. Ce modèle biomécanique intègre des matériaux hyperélastiques complexes permettant de capturer la déformation et les propriétés mécaniques réalistes du foie, essentielles pour assurer un recalage durant l'intervention. Le plugin SOniCS a été conçu pour offrir une compréhension intuitive des systèmes biomécaniques complexes, tout en permettant aux utilisateurs d'explorer facilement différents choix de modèles hyperélastiques sans nécessiter une expertise approfondie. Ainsi, les modèles de comportement des tissus peuvent être modifiés en une seule ligne de code. Cette simplicité dans la construction de nouveaux modèles fait de SOniCS un outil pour développer des modèles réduits ou de substitution et pour entraîner des algorithmes d'apprentissage automatique, en vue de simulations en temps réel spécifiques au patient.

Une description du fonctionnement de SOniCS est illustrée sur la figure 9.2. Dans SOFA, chaque élément doit être défini, incluant la géométrie de la cellule, les fonctions de forme (y compris leurs dérivées), ainsi que le schéma et le degré de la quadrature. Dans SOniCS, cela a été remplacé par seulement deux lignes de code Python pour décrire l'élément et sa quadrature. Le même avantage s'applique pour la description du modèle de matériau. Dans SOFA, chaque matériau doit être créé dans un fichier séparé, précisant son énergie de déformation, en dérivant manuellement le second tenseur de Piola-Kirchhoff ($\boldsymbol{S}$) et sa Jacobienne. Cela a été remplacé dans SOniCS par une simple définition de l'énergie de déformation du modèle de matériau souhaité en UFL. La dérivée de l'énergie de déformation sera ensuite calculée automatiquement en utilisant le module FFCX. Enfin, les deux plugins partagent les mêmes méthodes Forcefield pour assembler le vecteur résiduel global ($\boldsymbol{R}$) et la matrice de rigidité ($\boldsymbol{K}$).

Nous avons validé SOniCS à l'aide de simulations numériques, y compris des solutions manufacturées, des simulations de poutres en porte-à-faux et des benchmarks fournis par FEBio. Nous avons ainsi atteint une précision de l'ordre de celle de la machine et démontré l'utilisation de SOniCS dans une simulation haptique en temps réel, où un instrument chirurgical contrôlé par l'utilisateur interagit avec un foie hyperélastique. Enfin, des exemples complets de l'utilisation de notre plugin pour des simulations reposant

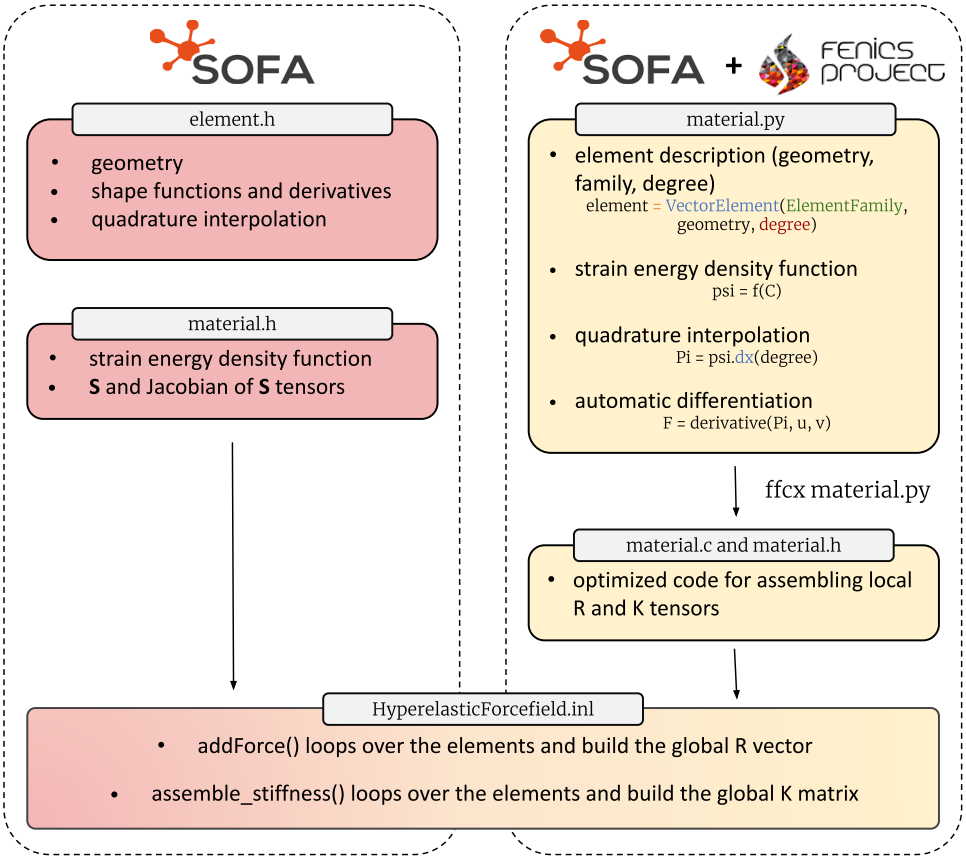


Figure 9.2: Description du pipeline SOniCS (à droite) et des différences avec SOFA (à gauche)

sur les modèles de Saint Venant-Kirchhoff, Neo-Hookean, Mooney-Rivlin et Holzapfel-Ogden sont fournis en tant que matériel complémentaire. Pour valider SOniCS, nous avons mené des benchmarks en comparant ses résultats avec ceux de SOFA sur un cas de test classique : la déflexion d'une poutre en porte-à-faux. Ce cas permet de raffiner facilement le maillage en raison de la simplicité de la géométrie et de modifier les conditions aux limites pour simuler des expériences réelles. Dans ce cadre, la poutre était fixée du côté droit (condition de Dirichlet naturelle sur Γ_D), tandis que des conditions aux limites de type Neumann étaient appliquées du côté gauche (Γ_N). Nous avons modélisé la déformation de la poutre avec deux modèles hyperélastiques : Saint Venant-Kirchhoff et Neo-Hookéen, en fixant les paramètres mécaniques et les conditions aux limites de Neumann égales à $-10\,\text{Pa}$ dans la direction y, jusqu'à ce que des déformations suffisantes soient atteintes, avec les paramètres suivants : $E = 3\,\text{kPa}$ et $\nu = 0.3$.

L'objectif de cette étude était de comparer les solutions obtenues par éléments finis

avec SOniCS et SOFA, sous les mêmes contraintes, en termes de performances de calcul et de temps d'exécution. Pour ce faire, nous avons calculé l'erreur relative L^2 pour les champs de déplacement et de déformation ($L^2_u(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{SOFA}})$ et $L^2_E(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{SOFA}})$) entre la solution SOniCS et la solution de référence obtenue avec SOFA. Les résultats sont présentés dans les tableaux 9.1 et 9.2 pour les matériaux Saint Venant-Kirchhoff et Neo-Hookéen, respectivement.

Modèle Saint Venant-Kirchhoff					
Element	Nombres de DOFs	SOniCS temps moyen d'itération NR (s)	SOFA temps moyen d'itération NR (s)	$L^2_u(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{SOFA}})$	$L^2_E(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{SOFA}})$
P1	10935	0.387	0.438	4.10e-14	2.32e-16
P2	12705	0.810	0.808	3.49e-14	8.18e-15
Q1	10935	0.449	0.464	6.19e-13	3.76e-16
Q2	11772	0.839	0.942	3.81e-13	23.68e-14

Table 9.1: Erreur relative pour le déplacement ($L^2_u(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{SOFA}})$) et la déformation ($L^2_E(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{SOFA}})$), ainsi que le temps moyen d'itération NR (Newton-Raphson) entre SOniCS et SOFA pour différentes géométries d'éléments et schémas d'interpolation, en utilisant le modèle de matériau Saint Venant-Kirchhoff. Les éléments P1 et P2 correspondent à des tétraèdres linéaires ou quadratiques, tandis que les éléments Q1 et Q2 désignent des cubes linéaires et quadratiques.

Nous avons réalisé plusieurs expériences numériques afin de développer une compréhension intuitive des nouveaux modèles matériels dans SOFA en utilisant le plugin SOniCS. Tout d'abord, nous avons validé les modèles hyperélastiques les plus courants : Saint Venant-Kirchhoff et Neo-Hookean à l'aide d'une solution théorique. Ensuite, en utilisant FEBio comme référence, nous avons vérifié notre implémentation d'un modèle de matériau Mooney-Rivlin en utilisant des éléments P1, P2, Q1 et Q2. L'application finale a utilisé un dispositif haptique pour interagir en temps réel avec un modèle de foie anisotrope de Holzapfel Ogden. L'étude a utilisé notre plugin SOniCS pour générer du code C optimisé pour des modèles matériels complexes compatibles avec SOFA. D'une part, nous avons bénéficié des capacités de différentiation automatique et de génération de code de FEniCS pour contourner les difficultés liées à la dérivation et à l'implémentation du jacobien consistant dans SOFA. D'autre part, nous avons mis en œuvre des champs de force SOFA compatibles et conviviaux pour utiliser les noyaux C de FEniCS. Un utilisateur de SOFA peut désormais définir facilement un nouveau modèle matériel en spécifiant

Modèle Neo-Hookean					
Element	Nombres de DOFs	SOniCS temps moyen d'itération NR (s)	SOFA temps moyen d'itération NR (s)	$L^2_u(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{SOFA}})$	$L^2_E(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{SOFA}})$
P1	10935	0.391	0.428	2.17e-14	7.78e-14
P2	12705	0.826	0.852	1.40e-14	2.18e-20
Q1	10935	0.471	0.478	4.92e-13	5.77e-16
Q2	11772	0.826	0.864	1.64e-13	2.17e-14

Table 9.2: Erreur relative pour le déplacement ($L^2_u(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{SOFA}})$) et la déformation ($L^2_E(\boldsymbol{u}_{\texttt{SOniCS}}, \boldsymbol{u}_{\texttt{SOFA}})$), ainsi que le temps moyen d'itération NR (Newton-Raphson) entre SOniCS et SOFA pour différentes géométries d'éléments et schémas d'interpolation, en utilisant le modèle de matériau Neo-Hookéen. Les éléments P1 et P2 correspondent à des tétraèdres linéaires ou quadratiques, tandis que les éléments Q1 et Q2 désignent des cubes linéaires et quadratiques.

uniquement sa fonction d'énergie de déformation, la géométrie de l'élément ou la famille, ainsi que le schéma et le degré de quadrature en Python. Le code open-source, ainsi que toutes les données et cas de test, sont disponibles en tant que matériel complémentaire. Dans le cadre de cette thèse, SOniCS a permis de créer des modèles biomécaniques précis du foie, essentiels pour la simulation réaliste des déformations organiques. Ces modèles sont basés sur des matériaux hyperélastiques complexes, adaptés à la mécanique du foie, et ont été utilisés pour effectuer un recalage des données intraopératoires obtenues par IVUS.

9.3 La reconstruction de volume IVUS

En général, les volumes d'échographie sont reconstruits en suivant une séquence d'images, en les positionnant dans l'espace en fonction de leurs données de suivi correspondantes, puis en interpolant entre elles. Des systèmes de suivi électromagnétique sont souvent utilisés pour cette tâche. Cependant, leur utilisation en chirurgie laparoscopique est limitée en raison de contraintes de taille. De plus, ces systèmes peuvent introduire de la complexité et augmenter le coût de l'installation chirurgicale, tandis que leur précision peut être compromise par la présence d'instruments métalliques dans l'environnement chirurgical. Dans cette section, nous abordons le défi de la reconstruction des volumes IVUS sans recourir à des systèmes de suivi. Cette reconstruction de volume est essen-

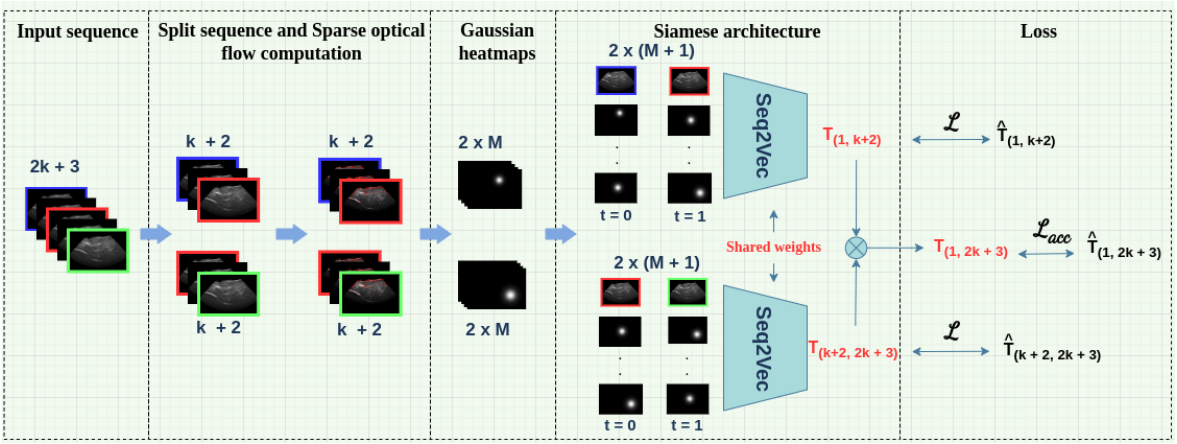


Figure 9.3: Vue d'ensemble de la méthode proposée

tielle dans le cadre de cette thèse, car elle constitue les données 3D intraopératoires. Une fois segmenté, le volume IVUS fournit des caractéristiques intraopératoires clés, cruciales pour la méthode de recalage basée sur les caractéristiques utilisée dans cette thèse.

Nous proposons une méthode de reconstruction de volumes échographiques sans suivi, spécifiquement adaptée à la chirurgie mini-invasive. Cette approche repose sur une architecture de type Siamese, incluant un réseau neuronal récurrent qui exploite les caractéristiques des images échographiques et le flux optique pour estimer la position relative des différentes images. Notre méthode ne nécessite aucun capteur additionnel, ce qui la rend particulièrement adaptée aux environnements chirurgicaux où l'espace et les ressources sont limités. La figure 9.3 présente un aperçu général de notre méthode, illustrant les principales étapes de la reconstruction du volume échographique sans suivi. La séquence d'entrée est divisée en deux sous-séquences égales, partageant une image commune. Ces deux séquences sont utilisées pour calculer un flux optique sparse. Des cartes thermiques gaussiennes suivant M points sont ensuite combinées avec la première et la dernière image de chaque sous-séquence pour former l'entrée du réseau. Nous utilisons une architecture Siamese basée sur un réseau Sequence to Vector (Seq2Vec). L'apprentissage se fait en minimisant l'erreur quadratique moyenne entre la sortie et les transformations de vérité terrain.

La méthode a été évaluée sur des données ex vivo provenant de porcs, obtenant des erreurs de translation et d'orientation respectivement de 0.449 ± 0.189 mm et 1.3 ± 1.5 degrés pour l'estimation de la position relative. En dépit des mouvements de rotation prédominants dans notre contexte, notre méthode parvient à une bonne reconstruction, avec un taux de dérive final et moyen de 23.11% et 28.71% respectivement. La figure 9.4 montre la reconstruction du volume de deux séquences de tailles différentes avec notre méthode en rouge, comparée la reconstruction vérité terrain. Malgré le mouvement de rotation de la

sonde, les résultats d'estimation de la position relative obtenus par notre méthode restent très précis. Toutefois, on peut noter que l'erreur de dérive augmente avec la longueur de la séquence. Cela demeure un défi pour la communauté, même dans le cas de mouvements de translation de la sonde. Ce travail constitue, à notre connaissance, la première tentative de reconstruction de volumes dans le contexte de l'échographie intravasculaire, un système complexe où les techniques de suivi traditionnelles, souvent basées sur des systèmes électromagnétiques, ne sont pas toujours applicables.

(a) Séquence de 50 images **(b)** Séquence de 300 images

Figure 9.4: La reconstruction de deux séquences de longueurs respectives de 50 et 300 avec notre méthode en rouge, comparée aux séquences de vérité terrain.

9.4 Une réalité augmentée guidée par les vaisseaux

Après avoir mis en place un modèle biomécanique du foie à partir des données préopératoires et reconstruit un volume à partir des données IVUS intraopératoires, nous nous concentrons désormais sur le recalage entre ces données préopératoires et intraopératoires. Cette étape est cruciale pour garantir une précision optimale lors de l'intégration des informations dans un système de RA pour la chirurgie. De nombreuses études ont démontré les avantages de l'intégration de la RA dans les salles d'opération pour assister les chirurgies du foie. Cette RA repose sur un recalage efficace entre les données préopératoires et intraopératoires. Cette thèse considère l'IVUS comme modalité intraopératoire pour plusieurs raisons, détaillées plus haut. Une vue d'ensemble de notre méthode est illustrée sur la figure 9.5. Nous proposons une méthode de recalage non rigide spécifique au patient, utilisant des modèles biomécaniques en raison de leur capacité à extrapoler à partir de données rares et bruitées, comme mentionné précédemment. À partir d'un scanner CT préopératoire, nous segmentons le foie et ses structures internes pour créer un jumeau numérique spécifique au patient à l'aide de modèles biomécaniques. Ce jumeau numérique est utilisé pour simuler les données IVUS en introduisant des déformations, en supprimant des branches et en ajoutant du bruit à l'arbre vasculaire. Grâce

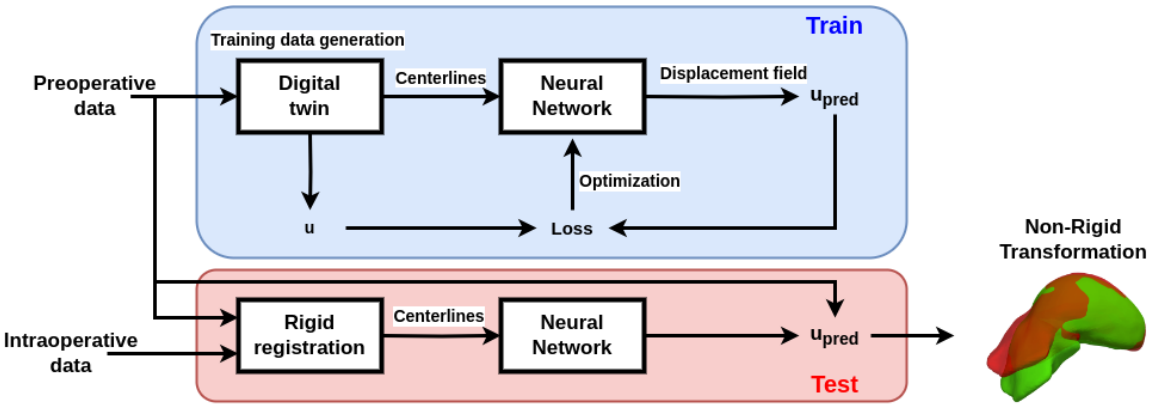


Figure 9.5: Un jumeau numérique du foie est utilisé pour entraîner un réseau neuronal sur la tâche de recalage non rigide. Le recalage rigide en faisant un appariement entre les branches correspondantes des arbres vasculaires provenant des deux modalités.

à ce processus, nous générons un ensemble de données d'entraînement et de validation. Pour le recalage non rigide, nous utilisons une architecture U-Net pour apprendre la correspondance entre les lignes centrales de l'arbre vasculaire intraopératoire et le champ de déplacement correspondant, permettant ainsi un recalage non rigide entre le modèle préopératoire et l'arbre vasculaire intraopératoire. De plus, nous exploitons la structure en graphes de l'arbre vasculaire pour appairer les veines porte principales dans leurs configurations préopératoire et intraopératoire, permettant ainsi un recalage rigide. Il est important de noter que nous validons notre méthode en utilisant des données IVUS simulées en raison de l'absence de jeux de données IVUS réels.

Pour évaluer notre méthode, nous avons segmenté le parenchyme hépatique, la veine porte et la tumeur à partir de données réelles de patients issues de l'étude de Soler et al. (2010). Nous avons construit un modèle biomécanique en prenant un module de Young de 4000 Pa et un coefficient de Poisson de 0,4. Ensuite, nous avons généré des jeux de données d'entraînement et de validation. Les forces appliquées ont été échantillonnées à partir d'une distribution uniforme sur l'intervalle [0, 30] N, afin de couvrir des déformations organiques à la fois petites et grandes. Le jeu de données généré contient 3600 déformations différentes, dont 3200 pour l'entraînement et 400 pour la validation. Nous avons considéré deux scénarios liés à l'acquisition IVUS. Dans le premier, l'arbre vasculaire complet est observé lors de l'acquisition, ce qui correspond à un balayage de 360° du cathéter IVUS. Le deuxième scénario correspond à un cas où seule une partie de l'arbre vasculaire est observée. Dans les deux cas, nous supposons que la veine porte principale est visible, car elle sert de référence pour le recalage rigide initial. Les sous-sections suivantes décrivent la validation de notre méthode pour ces deux scénarios, en utilisant le modèle biomécanique comme base et en évaluant la prédiction des réseaux à l'aide de

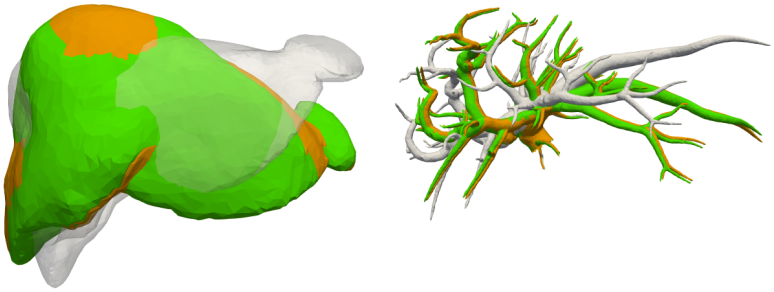


Figure 9.6: Caption

l'erreur de localisation de la tumeur (TRE). Les résultats du réseau pour le jeu de validation sont présentés dans le tableau. Dans le cas ou tout les vaisseaux sont visibles sur les données IVUS. Nous obtenons une erreur TRE moyenne de 0,6 mm avec une erreur maximale de 2,97 mm sur les 400 échantillons de validation. Nous rapportons également les erreurs relatives, qui correspondent à un pourcentage de la magnitude de la déformation pour chaque échantillon. Les prédictions sont calculées en 34 ms. La figure 9.6 montre les résultats d'un échantillon de validation, où l'on peut observer l'alignement entre la prédiction et la vérité terrain. En gris, l'état initial de l'organe (état préopératoire) ; en vert, la déformation de la vérité terrain (issue du modèle biomécanique) ; et en orange, la prédiction du réseau. De plus, nous avons comparé nos résultats à l'approche U-Mesh proposée par Brunet et al. (2019). Leur modalité intraopératoire est la vidéo laparoscopique, où un nuage de points 3D de la surface de l'organe est reconstruit à partir des images vidéo. Nous avons utilisé leur implémentation pour synthétiser les nuages de points et entraîner et valider le réseau. Voir le tableau 9.3.

En utilisant le même processus que pour le scénario précédent, nous avons généré un jeu de données correspondant à un scénario dans lequel seul un arbre vasculaire partiel est visible lors de l'acquisition IVUS. Dans ce scénario, nous avons considéré deux cas où respectivement 50% et 30% de l'arbre vasculaire sont observés. Dans les deux cas, les branches vasculaires sont choisies dans les zones où se trouvent les tumeurs. En effet, les chirurgiens savent dans quel segment anatomique du foie les tumeurs sont localisées et acquièrent généralement des données dans ces zones. Un jeu de données a été généré pour chacun de ces scénarios et le réseau a été entraîné sur les 3200 échantillons d'entraînement. Les résultats du réseau sur le jeu de validation de 400 échantillons sont rapportés dans le tableau 9.3. Les résultats montrent que le réseau reste performant en termes d'erreur TRE au niveau des localisations des tumeurs. Nous obtenons une TRE moyenne de 0,59 mm et 0,63 mm pour les cas respectifs de 50% et 30% d'arbre vasculaire intraopératoire, avec une TRE maximale de 1,28 mm et 2,36 mm sur les 400 échantillons de validation. Le temps de calcul est de 34 ms.

	$e_{mean}(mm)$	$e_{max}(mm)$	$e^{relative}_{mean}(\%)$	$e^{relative}_{max}(\%)$
U-Mesh [Brunet et al. (2019)]	0.83 ± 0.42	3.01	0.22 ± 0.11	0.78
Ours (30%)	0.59 ± 0.37	2.36	0.16 ± 0.10	0.63
Ours (50%)	0.63 ± 0.24	1.28	0.17 ± 0.06	0.33
Ours (full)	0.60 ± 0.37	2.97	0.16 ± 0.09	0.81

Table 9.3: Les résultats de notre méthode en termes d'erreurs TRE pour les scénarios : arbre vasculaire complet, partiel (50%) et partiel (30%). Les erreurs sont calculées en comparant la prédiction du réseau avec la déformation du modèle biomécanique pour la même déformation intraopératoire.

9.5 Conclusion

Cette thèse vise à développer un nouveau système de réalité augmentée (RA) guidé par une sonde d'échographie intravasculaire (IVUS) pour les interventions hépatiques mini-invasives. En effet, malgré les avantages reconnus de l'échographie IVUS pour fournir des images peroperatoires, à notre connaissance, aucune méthode n'a été proposée jusqu'à présent pour intégrer les données IVUS dans les systèmes de guidage chirurgical pour la chirurgie du foie. Le système développé dans cette thèse vise à améliorer la précision chirurgicale en intégrant les données IVUS en temps réel avec les images préopératoires, offrant ainsi aux chirurgiens une vue augmentée des structures internes du foie. Cela peut être réalisé en reconstituant et en recalant les données IVUS avec des scans CT préopératoires, fournissant ainsi une visualisation anatomiquement précise et à jour pendant l'opération, qui s'adapte à la déformation de l'organe. Dans la première partie de ce travail, nous avons présenté une revue des méthodes existantes de recalage, en nous concentrant sur les techniques basées sur l'intensité et les caractéristiques. Bien que le recalage basé sur l'intensité soit souvent utilisé en imagerie médicale, il pose des défis dans les scénarios de multimodalité, tels que le recalage des données IVUS avec le CT. Par conséquent, nous avons utilisé dans cette thèse des méthodes de recalage basées sur les caractéristiques. Associées à des modèles biomécaniques, ces approches permettent de résoudre le problème de recalage en évitant les difficultés liées à la multimodalité et en exploitant la capacité des modèles biomécaniques à extrapoler à partir de données peropératoires bruitées, telles que celles provenant de l'IVUS.

L'une des principales contributions techniques de cette thèse est le développement de SOniCS, une méthode qui simplifie la mise en œuvre de modèles de matériaux hyperélastiques pour simuler les tissus mous. En intégrant FEniCS, un outil pour résoudre les équations aux dérivées partielles, avec SOFA, un logiciel pour la simulation en temps

réel d'objets déformables, SOniCS permet une modélisation efficace et précise du comportement complexe des tissus. Celui-ci a permis le developpement d'un modèle biomécanique du foie avec les données préopératoires. Ce modèle biomécanique constitue la base nécessaire pour recaler les données IVUS peropératoires, assurant un alignement précis et une adaptabilité aux conditions peropératoires.

Nous avons également relevé le défi de la reconstruction du volume IVUS sans avoir recours à des systèmes de capteursexternes. Les systèmes de capteurs traditionnels sont limités dans les chirurgies mini-invasives en raison de contraintes de taille. En outre, ils peuvent être coûteux et sujets aux interférences des instruments chirurgicaux. Pour surmonter ces limitations, nous avons développé une nouvelle méthode basée sur l'intelligence artificielle pour reconstruire des volumes 3D IVUS, éliminant ainsi la nécessité de systèmes de suivi basés sur des capteurs externes. Cette approche simplifie la configuration chirurgicale, réduit les coûts du système peropératoire, en le rendant plus pratique pour une utilisation en conditions réelles. Enfin, la contribution majeure de cette thèse est le développement d'une méthode pour le recalage en temps réel des volumes IVUS peropératoires avec les données CT préopératoires. Cette méthode s'appuie sur l'apprentissage profond, plus précisément un réseau de neurones entraîné sur des données générées par le modèle biomécanique du foie, afin d'aligner les données pré et intra opératoires. Cela permet des ajustements dynamiques du modèle préopératoire pendant la chirurgie. Nous avons également étendu ce cadre de recalage pour l'adapter aux variations spécifiques à chaque patient. En reconnaissant que les modèles biomécaniques reposent souvent sur des hypothèses concernant les propriétés matérielles et les conditions aux limites qui peuvent varier entre les patients, nous avons amélioré celui-ci en ayant une valeur précise des characteristiques du patient. Ce processus intègre une boucle d'optimisation qui identifie les propriétés matérielles et les conditions aux limites correspondant le mieux aux observations intraopératoires. Cela permet au modèle biomécanique de s'adapter dynamiquement aux caractéristiques mécaniques uniques de chaque patient.

APPENDIX A

INTRAOPERATIVE CT AUGMENTATION FOR NEEDLE-BASED LIVER INTERVENTIONS

This appendix presents work conducted at the very beginning of my PhD, at a time when I had no Ultrasound data related to the main subject of this thesis. It focuses on improving CT guidance during needle-based liver procedures, such as tumor ablation, while minimizing the need for contrast agent injections. The approach involves augmenting intraoperative CT scans by integrating the preoperative vascular network, which is deformed to align with the current scan. The augmented CT is generated by fusing this vascular network with the non-contrasted intraoperative CT. Although this work is not directly related to the core topic of my PhD, it contributes in advancing RFA intervention accuracy. It led to the following publication:

- **S. El hadramy**, J. Verde, N. Padoy, S. Cotin. In: Greenspan, H., et al. "Intraoperative CT augmentation for needle-based liver interventions". *Medical Image Computing and Computer Assisted Intervention – MICCAI 2023*. Lecture Notes in Computer Science, vol 14228 [El hadramy et al. (2023b)].

A.1 Introduction

Needle-based liver tumor ablation techniques (e.g., radiofrequency, microwave, laser, cryoablation) have a great potential for local curative tumor control [Izumi et al. (2001)], with comparable results to surgery in the early stages for both primary and secondary cancers. Furthermore, as it is minimally invasive, it has a low rate of major complications and procedure-specific mortality and is tissue-sparing. Thus, its indications are growing exponentially and extending the limits to more advanced tumors [Schullian et al. (2020)]. CT guidance is a widely used imaging modality for placing needles, monitoring the treatment, and following up with patients. However, it is limited by exposure to ionizing radiation and the need for intravenous injection of contrast agents to visualize the intrahepatic vessels and the target tumor(s).

In standard clinical settings, the insertion of each needle requires multiple checkpoints during its progression, fine-tuning maneuvers, and eventual repositioning. This leads to multiple CT acquisitions to control the progression of the needle with respect to the vessels, the target, and other sensible structures [Arnolli et al. (2018)]. However, intrahepatic vessels (and some tumors) are only visible after contrast enhancement, which has a short lifespan and dose-related deleterious kidney effects. It makes it impossible to perform each control CT acquisition under contrast injection. A workaround to shortcut these limitations is to perform an image fusion between previous contrasted and intraoperative non-contrasted images. However, such a solution is only available in a limited number of clinical settings, and the registration is only rigid, usually deriving in bad re-

sults. In this work, we propose a method for visualizing intrahepatic structures after organ motion and needle-induced deformations in non-injected images by exploiting image features that are generally not perceivable by the human eye in common clinical workflows.

Two main strategies could be considered to address this challenge: **image fusion** and **image processing** techniques. Image fusion typically relies on the estimation of rigid or non-rigid transformations between 2 images to bring into the intraoperative image structures of interest only visible in the preoperative data. This process is often described as an optimization problem [Avants et al. (2011); Klein et al. (2010)], which can be computationally expensive when dealing with non-linear deformations, making their use in a clinical workflow limited. Recent deep learning approaches [Cao et al. (2018); Sokooti et al. (2017); Yang et al. (2017)] have proved to be a successful alternative to solve image fusion problems, even when a large non-linear mapping is required. When ground-truth displacement fields are not known, state-of-the-art methods use unsupervised techniques, usually an encoder-decoder architecture [Balakrishnan et al. (2019)], to learn the unknown displacement field between the 2 images. However, such unsupervised methods fail at solving our problem due to the lack of similar image features between the contrasted (CCT) and non-contrasted (NCCT) image in the vascular tree region (see section A.4).

On the other hand, deep learning techniques have proven to be very efficient at solving image processing challenges [Suganyadevi et al. (2022)]. For instance, image segmentation [Ramesh et al. (2018)], image style transfer [Gatys et al. (2015)], or contrast-enhancement to cite a few. Yet, segmenting vessels from non-contrasted images remains a challenge for the medical imaging community[Ramesh et al. (2018)]. Style transfer aims to transfer the style of one image to another while preserving its content [Gatys et al. (2015)]. However, applying such methods to generate a contrasted intraoperative CT is not a sufficiently accurate solution for the problem that we address. Contrast-enhancement methods could be an alternative. In the method proposed by Seo *et al.* [Seo et al. (2021)], a deep neural network synthesizes contrast-enhanced CT from non-contrast-enhanced CT. Nevertheless, results obtained by this method are not sufficiently robust and accurate to provide an augmented intraoperative CT on which needle-based procedures can be guided.

In this work, we propose an alternative approach, where a neural network learns local image features in a NCCT image by leveraging the known preoperative vessel tree geometry and topology extracted from a matching (undeformed) CCT. Then, the augmented CT is generated by fusing the deformed vascular tree with the non-contrasted intraoperative CT. Section A.2 presents the method and its integration in the medical workflow. Section

A.3 presents and discusses the results, and finally we conclude in section A.5 and highlight some perspectives.

A.2 Method

In this section, we present our method and its compatibility with current clinical workflows. A few days or a week before the intervention, a preoperative diagnostic multiphase contrast-enhanced image (MPCECT) is acquired (Fig. A.1, yellow box). The day of the intervention, a second MPCECT image is acquired before starting the needle insertion, followed by a series of standard, non-injected acquisitions to guide the needle insertion (Fig. A.1, blue box). Using such a non-contrasted intraoperative image as input, **our method performs a combined non-rigid registration and augmentation of the intraoperative CT** by adding anatomical features (mainly intrahepatic vessels and tumors) from the preoperative image to the current image. To achieve this result, our method only requires to process and train on the baseline MPCECT image (Fig. A.1, red box). An overview of the method is shown in the Fig A.2 and the following sections describe its main steps.

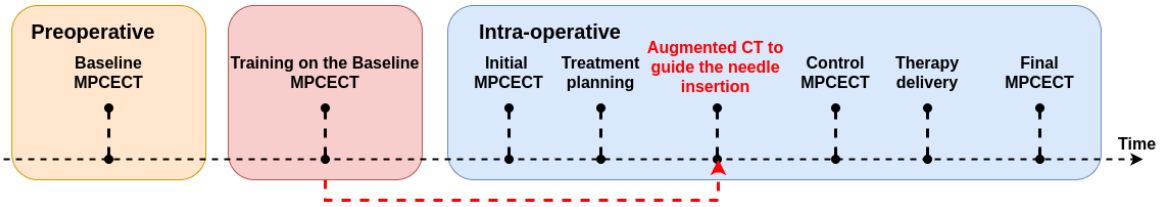


Figure A.1: Integration of our method in the clinical workflow. The neural network trained on preoperative MPCECT avoids contrast agent injections during the intervention.

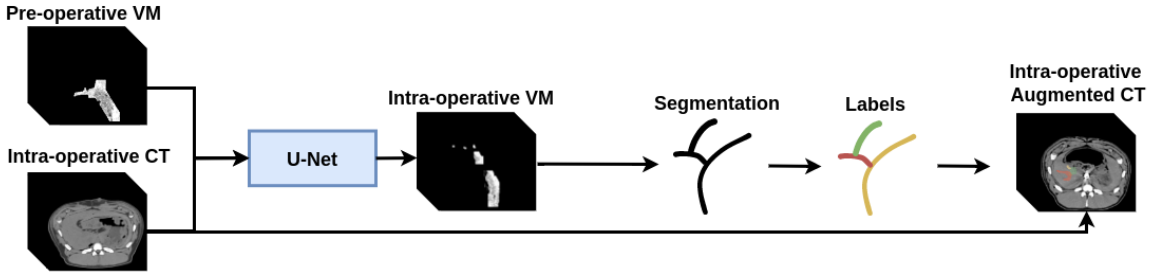


Figure A.2: The neural network takes as input the preoperative vessel map (VM) and the intraoperative NCCT, and outputs the intraoperative vessel map (VM) from which we extract the deformed vascular tree. Finally, the augmented CT is created by fusing the segmented image and labels with the intraoperative NCCT.

A.2.1 Vessel map extraction

We call Vessel Map (VM) the region of interest defining the vascular tree in the NCCT. Since vascular structures are not visible in non-contrasted images, the extraction of this map is done by segmenting the CCT and then using this segmentation as a mask in the NCCT. Mathematical morphology operators, in particular a dilation operation [Haralick et al. (1987)], are performed on the segmented region of interest to slightly increase its dimensions. This is needed to compensate for segmentation errors and the slight anatomical motion that may exist between the contrasted and non-contrasted image acquisitions. In practice, the acquisition protocols limit the shift between the NCCT and CCT acquisitions, and only a few sequential dilation operations are needed to ensure we capture the true vessel fingerprint in the NCCT image. Note that the resulting vessel map is not a binary mask but a subset of the image limited to the volume covered by the vessels.

A.2.2 Data augmentation

The preoperative MPCECT provides a couple of registered NCCT and CCT images. This is obviously not sufficient for training purposes, as they do not represent the possible soft tissue deformation that may occur during the procedure. Therefore, we augment the data set by applying multiple random deformations to the original images. Random deformations are created by considering a predefined set of control points for which we define a displacement field with a random normal distribution. The displacement field of the full volume is then obtained by linearly interpolating the control points' displacement field to the rest of the volume. All the deformations are created using the same number of control points and characteristics of the normal distributions.

A.2.3 Neural Network Architecture

Predicting the vascular tree location in the deformed intraoperative NCCT is done using a U-net [Ronneberger et al. (2015)] architecture. The neural network takes as input the preoperative vessel map and the intraoperative NCCT, and outputs the intraoperative vessel map. Our network learns to find the image features (or vessel fingerprint) present in the vessel map, in a given NCCT assuming the knowledge of its geometry, topology, and the distribution of contrast from the preoperative MPCECT. The architecture of our network is illustrated in figure A.3. It consists of a four-layer analysis (left side) and synthesis (right side) paths that provide a non-linear mapping between low-resolution input and output images. Both paths include four 3x3x3 unpadded convolutions, each followed by a Leaky Rectified Linear Unit (LeakyReLU) activation function. The analysis includes a 2x2x2 max

pooling with a stride of 1, while the synthesis follows each convolution by a 2x2x2 up-convolution with a stride of 1. Shortcut connections from layers of equal resolution in the analysis path provide the essential high-resolution features of the synthesis path. In the last layer, a 1x1x1 convolution reduces the number of output channels to one, yielding the vessel map in the intraoperative image. Our network is trained by minimizing the mean square error between the predicted and ground truth vessel map. Training details are presented in section 3.

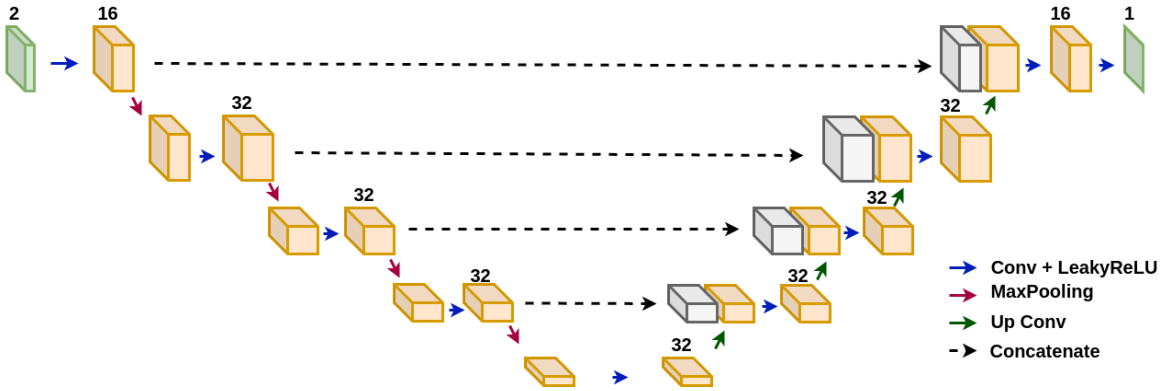


Figure A.3: Our neural network uses a four-path encoder-decoder architecture and takes as input a two-channel image corresponding to the intraoperative NCCT image concatenated with the preoperative vessel map. The output is the intraoperative vessel map.

A.2.4 CT augmentation

Once the network has been trained on the patient-specific preoperative data, the next step is to augment and visualize the intraoperative NCCT. This is done in 3 steps:

- The dilatation operations introduced in section A.2.1 are not reversible (i.e. the segmented vessel tree cannot be recovered from the VM by applying the same number of erosion operations). Also, neighboring branches in the vessel tree could end up being fused, thus changing the topology of the vessel map. Therefore, to retrieve the correct segmented (yet deformed) vascular tree, we compute a displacement field between the preoperative and intraoperative VMs. This is done with the Elastix library [Klein et al. (2010)]. The resulting displacement field is applied on the preoperative segmentation to retrieve the intraoperative vessel tree segmentation. This is illustrated in figure A.4.

- The augmented image is obtained by fusing the predicted intraoperative segmentation with the intraoperative NCCT image. The augmented vessels are displayed in green to ensure the clinician is aware this is not a true CCT image (see Fig. A.5).

- It is also possible to add anatomical labels to the intraoperative augmented CT to further assist the clinician. To achieve this objective, we compute a graph data structure from the preoperative segmentation. We first extract the vessel centerlines as described in [Antiga et al. (2003)]. To define the associated graph structure, we start by selecting all branches with either no parent or no children. The branch with the highest radius is then selected as the root edge. An oriented graph is created using a Breadth First Search algorithm starting from the root edge. Nodes and edges correspond respectively to vessel tree bifurcations and branches. We use the graph structure to associate each anatomical label (manually defined) with a Strahler [Devroye et Kruszewski (1996)] graph ordering. The same process is applied to the predicted intraoperative segmentation. This makes it possible to correctly map the preoperative anatomical labels (e.g. vessel name) and display them on the augmented image.

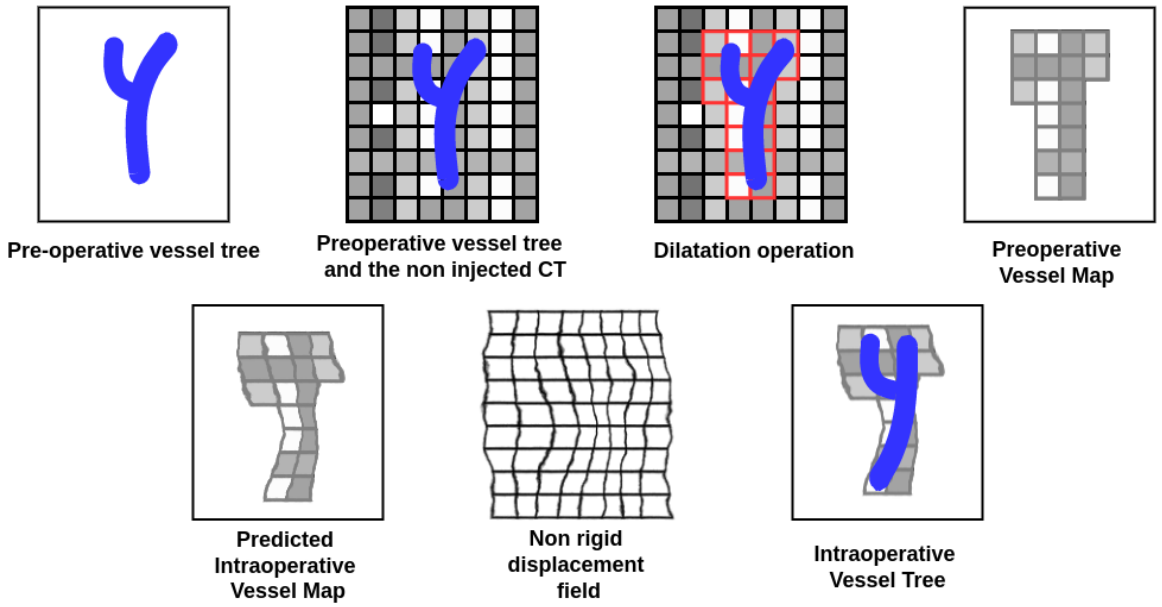


Figure A.4: This figure illustrates the different stages of the pipeline adopted to generate the VM and show how the vessel tree topology is retrieved from the predicted intraoperative VM by computing a displacement field between the preoperative VM and the predicted VM. This field is applied to the preoperative segmentation to get the intraoperative one.

A.3 Results and Discussion

A.3.1 Dataset and implementation details

To validate our approach, four couples of MPCECT abdominal porcine images were acquired from 4 different subjects. For a given subject, each couple corresponds to a pre-operative and an intraoperative MPCECT. We recall that an MPCECT contains a set of registered NCCT and CCT images. These images are then cropped and down-sampled to 256x256x256, and the voxels intensities are scaled between 0 and 255. Finally, we extracted the VM from each MPCECT sample and applied 3 dilation operations, which demonstrated the best performance in terms of prediction accuracy and robustness of our data. We note that public data sets such as DeepLesion [Yan et al. (2018)], 3Dircadb-01 [Soler et al. (2010)] and others do not fit our problem since they do not include the NCCT images. Aiming at a patient-specific prediction, we only train on a "subject" at a time. For a given subject, we generate 100 displacement fields using the data augmentation strategy explained above with 50 voxels for the control points spacing in the three spatial directions and a standard deviation of 5 voxels for the normal distributions. The resulting deformation is applied to the **preoperative** MPCECT and its corresponding VM. Thus, we end up with a set of 100 triplets (NCCT, CCT and VM). Two out of the 100 triplets are used for each training batch, where one is considered as the pre-operative MPCECT and the other as the intraoperative one. This makes it possible to generate up to 4950 training and validation samples. The **intraoperative** MPCECT of the same subject is used to test the network. Our method is implemented in Tensorflow 2.4, on a GeForce RTX 3090. We use an Adam optimizer ($\beta_1 = 0.001$, $\beta_2 = 0.999$) with a learning rate of 10^{-4}. The training process converges in about 1,000 epochs with a batch size of 1 and 200 steps per epoch.

A.3.2 Results

To assess our method, we use a dice score to measure the overlap between our predicted segmentation and the ground truth. Being a commonly used metric for segmentation problems, Dice aligns the nature of our problem as well as the clinical impact of our solution. We ha performed tests on 4 different (porcine) data sets. Results are reported in table A.1. The method achieved a mean dice score of 0.81. An example of a subject intraoperative augmented CT is illustrated in figure A.5, where the three images correspond respectively to the initial non injected CT, the augmented CT without and with labels. Figure A.6 illustrates the results of our method for *Subject 1*. The green vessels correspond to

Dice score	Subject 1	Subject 2	Subject 3	Subject 4	Mean	Std
Ours	**0.8**	**0.77**	**0.79**	**0.90**	**0.81**	**0.04**
Expert clinician	**0.52**	**0.45**	**0.53**	**0.52**	**0.51**	**0.03**

Table A.1: This table presents our results over 4 subjects in terms of dice score. For each subject the network was trained on the preoperative MPCECT and tested on the intraoperative MPCECT. We achieve a mean dice score of 0.81 vs. 0.51 for the clinical experts.

the ground truth intraoperative segmentation, the orange ones to the predicted intraoperative segmentation and finally the gray vessel tree corresponds to the preoperative CCT vessel tree. Such results demonstrate the ability of our method to perform very well even in the presence of large deformations.

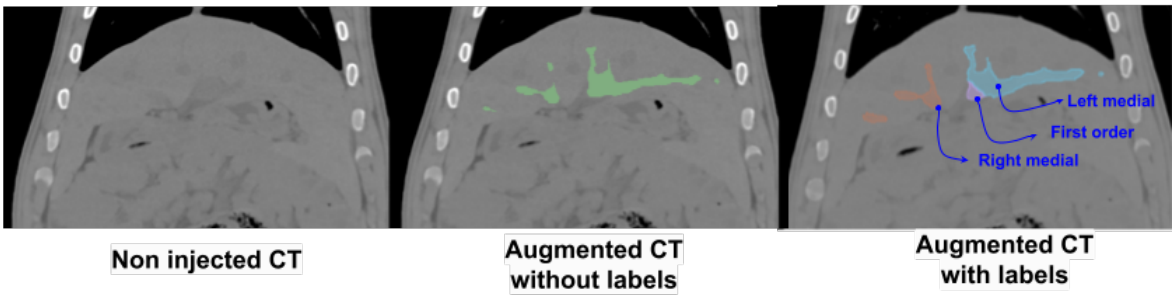


Figure A.5: In this figure we show the original NCCT on the left. The middle image shows the augmented CT with the predicted vessel tree (in green). The rightmost image shows the augmented image with anatomical labels transferred from the preoperative image segmentation and labelling.

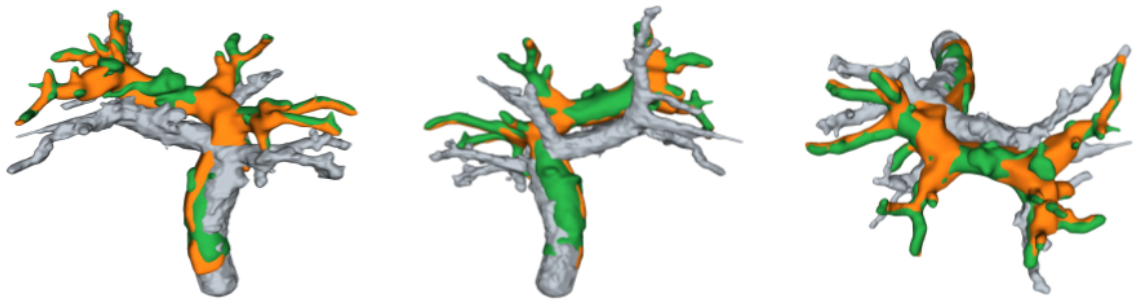


Figure A.6: Assessment of our method for Subject 1. We show 3 different views of the intraoperative vessel prediction (in orange), the ground truth (in green) and the preoperative vessels (in grey).

Qualitative assessment: To further demonstrate the value of our method, we have asked two clinicians to manually segment the NCCT images in the intraoperative MPCECT data. Their results (mean and standard deviation) are reported in table A.1. Our method out-

performs the results of both clinicians, with an average dice score of 0.81 against 0.51 as a mean for the clinical experts.

A.4 Ablation study and additional results

Vessel map: We have removed the VM from the network input to demonstrate its impact on our results. Using the data of the *Subject 1*, a U-net was trained to segment the vessel tree of the intraoperative NCCT image. The network only managed to segment a small portion of the main portal vein branch. Thus, achieving a dice score of **0.16** vs **0.79** when adding the preoperative VM as additional input. We also studied the influence of the diffusion kernel applied to the initial segmentation. We have seen, on our experimental data, that 3 dilation operations were sufficient to compensate for the possible motion between NCCT and CCT acquisitions.

Comparison with VoxelMorph: The problem that we address can be seen from different angles. In particular, we could attempt to solve it by registering the preoperative NCCT to the intraoperative one and then applying the resulting displacement field to the known preoperative segmentation. However, state-of-the-art registration methods such as VoxelMorph [Balakrishnan et al. (2019)] and others do not necessarily guarantee a diffeomorphic [Cao et al. (2005)] displacement field that ensures the continuity of the displacement field inside the parenchyma where the intensity is quite homogeneous on the NCCT. To assess this assumption, a VoxelMorph[1] network was trained on the *Subject 1* of our porcine data sets. We trained the network with both MSE and smoothness losses during 100 epochs and given a batch of size 4. Results are illustrated below in Figure A.7. While the VoxelMorph network accurately registers the liver shape, the displacement field is almost null in the region of vessels inside the parenchyma. Therefore, the preoperative vessel segmentation is not correctly transferred into the intraoperative image.

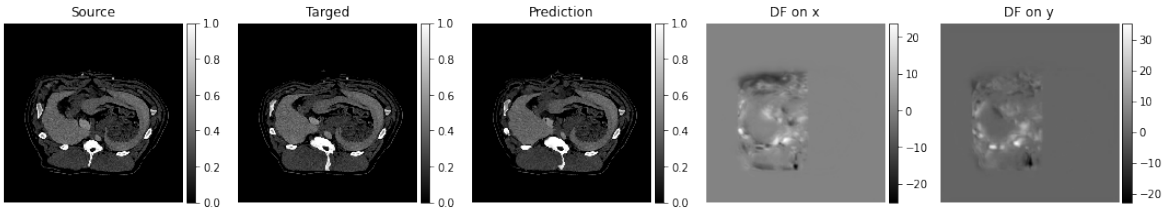


Figure A.7: Illustration of Voxelmorph registration between NCCT preoperative and intraoperative images. The prediction is the output of VoxelMorph method. DF stands for the displacement fields on x and y predicted by VoxelMorph method.

[1] https://github.com/voxelmorph/voxelmorph

A.5 Conclusion

In this work, we proposed a method for augmenting intra-operative NCCT images as a means to improve needle CT-guided techniques while reducing the need for contrast agent injection during tumor ablation procedures or other needle-based procedures. Our method uses a U-net architecture to learn local vessel tree image features in the NCCT by leveraging the known vessel tree geometry and topology extracted from a matching CCT image. The augmented CT is generated by fusing the predicted vessel tree with the NCCT. Our method is validated on several porcine images, achieving an average dice score of 0.81 on the predicted vessel tree location. In addition, it demonstrates robustness even in the presence of large deformations between the preoperative and intraoperative images. Our future steps will essentially involve applying this method to patient data and perform a small user study to evaluate the usefulness and limitations of our approach.

BIBLIOGRAPHY

Abdel-Basset, M., Fakhry, A. E., El-Henawy, I., Qiu, T., et Sangaiah, A. K. (2017). Feature and intensity based medical image registration using particle swarm optimization. *J. Med. Syst.*, 41(12):197.

Acidi, B., Ghallab, M., Cotin, S., Vibert, E., et Golse, N. (2023). Augmented reality in liver surgery. *J. Visc. Surg.*, 160(2):118–126.

Adams, R. B. (2014). Scanning techniques in transabdominal and Intraoperative/Laparoscopic ultrasound. In *Abdominal Ultrasound for Surgeons*, pages 31–60. Springer New York, New York, NY.

Ahrens, J., Geveci, B., et Law, C. (2005). Paraview: An end-user tool for large data visualization. *Visualization Handbook*.

Allard, J., Courtecuisse, H., et Faure, F. (2012). Chapter 21 - implicit fem solver on gpu for interactive deformation simulation. In mei W. Hwu, W., editor, *GPU Computing Gems Jade Edition*, Applications of GPU Computing Series, pages 281–294. Morgan Kaufmann, Boston.

Alnaes, M. S., Blechta, J., Hake, J., Johansson, A., Kehlet, B., A. Logg, C. R., Ring, J., Rognes, M. E., et Wells, G. N. (2015). The FEniCS project version 1.5. *Archive of Numerical Software*, 3.

Alnæs, M. S., Logg, A., Ølgaard, K. B., Rognes, M. E., et Wells, G. N. (2014). Unified Form Language: A Domain-specific Language for Weak Formulations of Partial Differential Equations. *ACM Trans. Math. Softw.*, 40(2):9:1–9:37.

Antiga, L., Ene-Iordache, B., et Remuzzi, A. (2003). Centerline computation and geometric analysis of branching tubular surfaces with application to blood vessel modeling. *International Conference in Central Europe on Computer Graphics and Visualization*.

Arbogast, T., Tao, Z., et Wang, C. (2022). Direct serendipity and mixed finite elements on convex quadrilaterals. *Numerische Mathematik*, 150(4):929–974.

Arnold, D. N. et Awanou, G. (2011). The Serendipity Family of Finite Elements. *Foundations of Computational Mathematics*, 11(3):337–344.

Arnolli, M. M., Buijze, M., Franken, M., de Jong, K. P., Brouwer, D. M., et Broeders, I. A. M. J. (2018). System for CT-guided needle placement in the thorax and abdomen: A design for clinical acceptability, applicability and usability. *Int. J. Med. Robot.*, 14(1):e1877.

Arruda, E. M. et Boyce, M. C. (1993). A three-dimensional constitutive model for the large stretch behavior of rubber elastic materials. *Journal of the Mechanics and Physics of Solids*, 41(2):389–412.

Arun, K. S., Huang, T. S., et Blostein, S. D. (1987). Least-squares fitting of two 3-D point sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-9(5):698–700.

Avants, B. B., Tustison, N. J., Song, G., Cook, P. A., Klein, A., et Gee, J. C. (2011). A reproducible evaluation of ANTs similarity metric performance in brain image registration. *Neuroimage*, 54(3):2033–2044.

Aylward, S. R., Jomier, J., Guyon, J.-P., et Weeks, S. (2003). Intra-operative 3D ultrasound augmentation. In *Proceedings IEEE International Symposium on Biomedical Imaging*, pages 421–424. IEEE.

Balakrishnan, G., Zhao, A., Sabuncu, M. R., Guttag, J., et Dalca, A. V. (2018). An unsupervised learning model for deformable medical image registration.

Balakrishnan, G., Zhao, A., Sabuncu, M. R., Guttag, J., et Dalca, A. V. (2019). VoxelMorph: A learning framework for deformable medical image registration. *IEEE Trans. Med. Imaging*, 38(8):1788–1800.

Balay, S., Abhyankar, S., Adams, M. F., Benson, S., Brown, J., Brune, P., Buschelman, K., Constantinescu, E., Dalcin, L., Dener, A., Eijkhout, V., Faibussowitsch, J., Gropp, W. D., Hapla, V., Isaac, T., Jolivet, P., Karpeev, D., Kaushik, D., Knepley, M. G., Kong, F., Kruger, S., May, D. A., McInnes, L. C., Mills, R. T., Mitchell, L., Munson, T., Roman, J. E., Rupp, K., Sanan, P., Sarich, J., Smith, B. F., Zampini, S., Zhang, H., Zhang, H., et Zhang, J. (2022). PETSc/TAO users manual. Technical Report ANL-21/39 - Revision 3.18, Argonne National Laboratory.

Baldi, P. (1995). Gradient descent learning algorithm overview: a general dynamical systems perspective. 6(1):182–195.

Banerjee, J., Sun, Y., Klink, C., Gahrmann, R., Niessen, W. J., Moelker, A., et van Walsum, T. (2019). Multiple-correlation similarity for block-matching based fast CT to ultrasound registration in liver interventions. *Med. Image Anal.*, 53:132–141.

Barnea, D. I. et Silverman, H. F. (1972). A class of algorithms for fast digital image registration. *IEEE Trans. Comput.*, C-21(2):179–186.

Baroli, D., Quarteroni, A., et Ruiz-Baier, R. (2012). Convergence of a stabilized discontinuous galerkin method for incompressible nonlinear elasticity. *Advances in Computational Mathematics*, 39(2):425–443.

Bastian, P., Blatt, M., Dedner, A., Dreier, N.-A., Engwer, C., Fritze, R., Gräser, C., Grüninger, C., Kempf, D., Klöfkorn, R., Ohlberger, M., et Sander, O. (2021). The Dune framework: Basic concepts and recent developments. *Computers & Mathematics with Applications*, 81:75–112.

Besl, P. J. et McKay, N. D. (1992). A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256.

Blau, R., Krivitsky, A., Epshtein, Y., et Satchi-Fainaro, R. (2016). Are nanotheranostics and nanodiagnostics-guided drug delivery stepping stones towards precision medicine? *Drug Resist. Updat.*, 27:39–58.

Boonvisut, P. et Cavuşoğlu, M. C. (2013). Estimation of soft tissue mechanical parameters from robotic manipulation data. *IEEE ASME Trans. Mechatron.*, 18(5):1602–1611.

Borazjani, I. (2013). Fluid–structure interaction, immersed boundary-finite element method simulations of bio-prosthetic heart valves. *Computer Methods in Applied Mechanics and Engineering*, 257:103–116.

Bordas, S., Nguyen, V. P., Dunant, C., Nguyen Dang, H., et Guidoum, A. (2006). An extended finite element library. *International Journal for Numerical Methods in Engineering*, 2:1–33.

Bouguet, J.-Y. (2001). Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5.

Bray, F., Ferlay, J., Soerjomataram, I., Siegel, R. L., Torre, L. A., et Jemal, A. (2018). Global cancer statistics 2018: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries. *CA Cancer J. Clin.*, 68(6):394–424.

Brunet, J.-N. (2020). *Exploring new numerical methods for the simulation of soft tissue deformations in surgery assistance*. Theses, Université de Strasbourg, 4 Rue Blaise Pascal.

Brunet, J.-N., Mendizabal, A., Petit, A., Golse, N., Vibert, E., et Cotin, S. (2019). Physics-based deep neural network for augmented reality during liver surgery. In *Lecture Notes in Computer Science*, Lecture notes in computer science, pages 137–145. Springer International Publishing, Cham.

Bui, H. P., Tomar, S., Courtecuisse, H., Cotin, S., et Bordas, S. P. A. (2018). Real-time error control for surgical simulation. *IEEE Transactions on Biomedical Engineering*, 65(3):596–607.

Campos, C., Elvira, R., Rodriguez, J. J. G., M. Montiel, J. M., et D. Tardos, J. (2021). ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM. *IEEE Trans. Robot.*, 37(6):1874–1890.

Cao, X., Yang, J., Wang, L., Xue, Z., Wang, Q., et Shen, D. (2018). Deep learning based inter-modality image registration supervised by intra-modality similarity. *Mach. Learn. Med. Imaging*, 11046:55–63.

Cao, Y., Miller, M. I., Winslow, R. L., et Younes, L. (2005). Large deformation diffeomorphic metric mapping of vector fields. *IEEE Trans. Med. Imaging*, 24(9):1216–1230.

Castaing, D., Kunstlinger, F., Habib, N., et Bismuth, H. (1985). Intraoperative ultrasonographic study of the liver. *Am. J. Surg.*, 149(5):676–682.

Chamberland, É., Fortin, A., et Fortin, M. (2010). Comparison of the performance of some finite element discretizations for large deformation elasticity problems. *Computers & Structures*, 88(11-12):664–673.

Chauhan, V. K., Zhou, J., Lu, P., Molaei, S., et Clifton, D. A. (2024). A brief review of hypernetworks in deep learning. *Artif. Intell. Rev.*, 57(9).

Chen, Y. et Medioni, G. (2002). Object modeling by registration of multiple range images. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 2724–2729 vol.3. IEEE Comput. Soc. Press.

Cherqui, D., Husson, E., Hammoud, R., Malassagne, B., Stéphan, F., Bensaid, S., Rotman, N., et Fagniez, P. L. (2000). Laparoscopic liver resections: a feasibility study in 30 patients. *Ann. Surg.*, 232(6):753–762.

Chinesta, F., Keunings, R., et Leygue, A. (2013). *The proper generalized decomposition for advanced numerical simulations*. SpringerBriefs in applied sciences and technology. Springer International Publishing, Cham, Switzerland, 2014 edition.

Collignon, A., Maes, F., Delaere, D., Vandermeulen, D., Suetens, P., et Marchal, G. (1995). Automated multi-modality image registration based on information theory.

Costa, K. D., Holmes, J. W., et Mcculloch, A. D. (2001). Modelling cardiac mechanical properties in three dimensions. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 359(1783):1233–1250.

Cothren, R. M., Shekhar, R., Tuzcu, E. M., Nissen, S. E., Cornhill, J. F., et Vince, D. G. (2000). Three-dimensional reconstruction of the coronary artery wall by image fusion of intravascular ultrasound and bi-plane angiography. *Int. J. Card. Imaging*, 16(2):69–85.

Cotin, S., Delingette, H., et Ayache, N. (1999). Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):62–73.

Courtecuisse, H., Allard, J., Duriez, C., et Cotin, S. (2010a). Asynchronous Preconditioners for Efficient Solving of Non-linear Deformations. In *VRIPHYS - Virtual Reality Interaction and Physical Simulation*, pages 59–68, Copenhagen, Denmark. Eurographics Association.

Courtecuisse, H., Allard, J., Kerfriden, P., Bordas, S., Cotin, S., et Duriez, C. (2013). Real-time simulation of contact and cutting of heterogeneous soft-tissues. *Medical image analysis*, 18:394–410.

Courtecuisse, H., Allard, J., Kerfriden, P., Bordas, S. P., Cotin, S., et Duriez, C. (2014). Real-time simulation of contact and cutting of heterogeneous soft-tissues. *Medical Image Analysis*, 18(2):394–410.

Courtecuisse, H., Jung, H., Allard, J., Duriez, C., Lee, D. Y., et Cotin, S. (2010b). Gpu-based real-time soft tissue deformation with cutting and haptic feedback. *Progress in Biophysics and Molecular Biology*, 103(2):159–168. Special Issue on Biomechanical Modelling of Soft Tissue Motion.

Curley, S. A., Izzo, F., Delrio, P., Ellis, L. M., Granchi, J., Vallone, P., Fiore, F., Pignata, S., Daniele, B., et Cremona, F. (1999). Radiofrequency ablation of unresectable primary and metastatic hepatic malignancies: results in 123 patients. *Ann. Surg.*, 230(1):1–8.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Math. Control Signals Systems*, 2(4):303–314.

Dagon, B., Baur, C., et Bettschart, V. (2008). A framework for intraoperative update of 3D deformable models in liver surgery. In *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE.

Dehghan, E., Lu, K., Yan, P., Tahmasebi, A., Xu, S., Wood, B. J., Abi-Jaoudeh, N., Venkatesan, A., et Kruecker, J. (2015). Surface-based registration of liver in ultrasound and CT. In Webster, R. J. et Yaniv, Z. R., editors, *Medical Imaging 2015: Image-Guided Procedures, Robotic Interventions, and Modeling*. SPIE.

DeSalvo, G. J. et Swanson, J. A. (1985). *ANSYS engineering analysis system users manual*. Swanson Analysis Systems, Houston, Pa.

Descottes, B., Glineur, D., Lachachi, F., Valleix, D., Paineau, J., Hamy, A., Morino, M., Bismuth, H., Castaing, D., Savier, E., Honore, P., Detry, O., Legrand, M., Azagra, J. S., Goergen, M., Ceuterick, M., Marescaux, J., Mutter, D., Hemptinne, B., Troisi, R., Weerts, J., Dallemagne, B., Jehaes, C., Gelin, M., Donckier, V., Aerts, R., Topal, B., Bertrand, C., Mansvelt, B., Krunckelsven, L., Herman, D., Kint, M., Totte, E., Schockmel, R., et Gigot, J. F. (2003). Erratum: Laparoscopic liver resection of benign liver tumors. *Surg. Endosc.*, 17(4):668–668.

Devroye, L. et Kruszewski, P. (1996). On the Horton-Strahler number for random tries. *Theor. Inform. Appl.*, 30(5):443–456.

Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Smagt, P. v. d., Cremers, D., et Brox, T. (2015). FlowNet: Learning optical flow with convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766. IEEE.

Dubey, S. R., Singh, S. K., et Chaudhuri, B. B. (2022). Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 503:92–108.

Dulucq, J. L., Wintringer, P., Stabilini, C., Berticelli, J., et Mahajna, A. (2005). Laparoscopic liver resections: a single center experience. *Surg. Endosc.*, 19(7):886–891.

Duriez, C. (2013). Control of elastic soft robots based on real-time finite element method. In *2013 IEEE International Conference on Robotics and Automation*, pages 3982–3987. 2013 IEEE International Conference on Robotics and Automation.

Duriez, C., Andriot, C., et Kheddar, A. (2004). A multi-threaded approach for deformable/-rigid contacts with haptic feedback. In *12th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2004. HAPTICS '04. Proceedings.*, pages 272–279.

Duriez, C., Dubois, F., Kheddar, A., et Andriot, C. (2006). Realistic haptic rendering of interacting deformable objects in virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 12(1):36–47.

Ehlers, W. et Markert, B. (2001). A linear viscoelastic biphasic model for soft tissues based on the theory of porous media. *Journal of biomechanical engineering*, 123:418–24.

El hadramy, S., Verde, J., Beaudet, K.-P., Padoy, N., et Cotin, S. (2023a). Trackerless volume reconstruction from intraoperative ultrasound images. In *Lecture Notes in Computer Science*, Lecture notes in computer science, pages 303–312. Springer Nature Switzerland, Cham.

El hadramy, S., Verde, J., Padoy, N., et Cotin, S. (2023b). Intraoperative CT augmentation for needle-based liver interventions. In *Lecture Notes in Computer Science*, Lecture notes in computer science, pages 291–301. Springer Nature Switzerland, Cham.

Elouneg, A., Sutula, D., Chambert, J., Lejeune, A., Bordas, S., et Jacquet, E. (2021). An open-source fenics-based framework for hyperelastic parameter estimation from noisy full-field data: Application to heterogeneous soft tissues. *Computers & Structures*, 255:106620.

Eppenhof, K. A. J. et Pluim, J. P. W. (2019). Pulmonary CT registration through supervised learning with convolutional neural networks. *IEEE Trans. Med. Imaging*, 38(5):1097–1105.

Faure, F., Duriez, C., Delingette, H., Allard, J., Gilles, B., Marchesseau, S., Talbot, H., Courtecuisse, H., Bousquet, G., Peterlik, I., et Cotin, S. (2012). SOFA: A Multi-Model Framework for Interactive Physical Simulation. In Payan, Y., editor, *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, volume 11 of *Studies in Mechanobiology, Tissue Engineering and Biomaterials*, pages 283–321. Springer.

Fedorov, A., Beichel, R., Kalpathy-Cramer, J., Finet, J., Fillion-Robin, J.-C., Pujol, S., Bauer, C., Jennings, D., Fennessy, F., Sonka, M., Buatti, J., Aylward, S., Miller, J. V., Pieper, S., et Kikinis, R. (2012). 3D slicer as an image computing platform for the quantitative imaging network. *Magn. Reson. Imaging*, 30(9):1323–1341.

Ferlay, J., Parkin, D. M., et Steliarova-Foucher, E. (2010). Estimates of cancer incidence and mortality in europe in 2008. *Eur. J. Cancer*, 46(4):765–781.

Feuerstein, M., Mussack, T., Heining, S. M., et Navab, N. (2007). Registration-free laparoscope augmentation for intra-operative liver resection planning. In Cleary, K. R. et Miga, M. I., editors, *Medical Imaging 2007: Visualization and Image-Guided Procedures*. SPIE.

Fu, Y., Lei, Y., Wang, T., Curran, W. J., Liu, T., et Yang, X. (2020). Deep learning in medical image registration: a review. *Phys. Med. Biol.*, 65(20):20TR01.

Fukuda, M., Mima, S., Tanabe, T., Haniu, T., Suzuki, Y., Hirata, K., et Terada, S. (1984). Endoscopic sonography of the liver–diagnostic application of the echolaparoscope to localize intrahepatic lesions. *Scand. J. Gastroenterol. Suppl.*, 102:24–38.

Galle, P. R., Forner, A., Llovet, J. M., Mazzaferro, V., Piscaglia, F., Raoul, J.-L., Schirmacher, P., et Vilgrain, V. (2018). EASL clinical practice guidelines: Management of hepatocellular carcinoma. *J. Hepatol.*, 69(1):182–236.

Gatys, L. A., Ecker, A. S., et Bethge, M. (2015). A neural algorithm of artistic style.

Geuzaine, C. et Remacle, J.-F. (2009). Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79:1309–1331.

Ghorbani, F., Shabanpour, J., Beyraghi, S., Soleimani, H., Oraizi, H., et Soleimani, M. (2021). A deep learning approach for inverse design of the metasurface for dual-polarized waves. *Appl. Phys. A Mater. Sci. Process.*, 127(11).

Gibbons, C. H. (1934). History of testing machines for materials. *Transactions of the Newcomen Society*, 15(1):169–184.

Gilles, B., Bousquet, G., Faure, F., et Pai, D. K. (2011). Frame-based elastic models. *ACM Trans. Graph.*, 30(2).

Goury, O., Amsallem, D., Bordas, S., Liu, W., et Kerfriden, P. (2016). Automatised selection of load paths to construct reduced-order models in computational damage micromechanics: from dissipation-driven random selection to bayesian optimization. *Computational Mechanics*, 58.

Guan, S.-Y., Wang, T.-M., Meng, C., et Wang, J.-C. (2018). A review of point feature based medical image registration. *Chin. J. Mech. Eng.*, 31(1).

Guo, G., Zou, Y., et Liu, P. X. (2021). A new rendering algorithm based on multi-space for living soft tissue. *Computers & Graphics*, 98:242–254.

Ha, D., Dai, A. M., et Le, Q. V. (2017). Hypernetworks. In *International Conference on Learning Representations*.

Han, L., Hipwell, J., Tanner, C., Taylor, Z., Mertzanidou, T., Cardoso, M. J., Ourselin, S., et Hawkes, D. (2011). Development of patient-specific biomechanical models for predicting large breast deformation. *Physics in medicine and biology*, 57:455–72.

Haouchine, N., Dequidt, J., Peterlik, I., Kerrien, E., Berger, M.-O., et Cotin, S. (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)*. IEEE.

Haralick, R. M., Sternberg, S. R., et Zhuang, X. (1987). Image analysis using mathematical morphology. *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-9(4):532–550.

Hauseux, P., Hale, J. S., et Bordas, S. P. (2017). Accelerating monte carlo estimation with derivatives of high-level finite element models. *Computer Methods in Applied Mechanics and Engineering*, 318:917–936.

Hauseux, P., Hale, J. S., Cotin, S., et Bordas, S. P. (2018). Quantifying the uncertainty in a hyperelastic soft tissue model with stochastic parameters. *Applied Mathematical Modelling*, 62:86–102.

He, Y. P. et Gu, L. X. (2011). Medical image registration using normal vector and intensity value. In *2011 International Conference on Intelligent Computation and Bio-Medical Instrumentation*. IEEE.

Henn, S. et Witsch, K. (2005). A variational image registration approach based on curvature scale space. In *Lecture Notes in Computer Science*, Lecture notes in computer science, pages 143–154. Springer Berlin Heidelberg, Berlin, Heidelberg.

Hill, D. L., Batchelor, P. G., Holden, M., et Hawkes, D. J. (2001). Medical image registration. *Phys. Med. Biol.*, 46(3):R1–45.

Hirose, O. (2021). A bayesian formulation of coherent point drift. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(7):2269–2286.

Hoffmann, K. R., Wahle, A., Pellot-Barakat, C., Sklansky, J., et Sonka, M. (1999). *Int. J. Card. Imaging*, 15(6):495–512.

Holzapfel, G. A. et Ogden, R. W. (2009). Constitutive modelling of passive myocardium: a structurally based framework for material characterization. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1902):3445–3475.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Netw.*, 4(2):251–257.

Hu, Y., Modat, M., Gibson, E., Ghavami, N., Bonmati, E., Moore, C. M., Emberton, M., Noble, J. A., Barratt, D. C., et Vercauteren, T. (2018). Label-driven weakly-supervised learning for multimodal deformarle image registration. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 1070–1074. IEEE.

Izumi, N., Asahina, Y., Noguchi, O., Uchihara, M., Kanazawa, N., Itakura, J., Himeno, Y., Miyake, S., Sakai, T., et Enomoto, N. (2001). Risk factors for distant recurrence of hepatocellular carcinoma in the liver after complete coagulation by microwave or radiofrequency ablation. *Cancer*, 91(5):949–956.

Jacquemin, T. et Bordas, S. P. A. (2021). A unified algorithm for the selection of collocation stencils for convex, concave, and singular problems. *International Journal for Numerical Methods in Engineering*, 122(16):4292–4312.

Jansari, C., Natarajan, S., Beex, L., et Kannan, K. (2019). Adaptive smoothed stable extended finite element method for weak discontinuities for finite elasticity. *European Journal of Mechanics - A/Solids*, 78:103824.

Jasak, H., Jemcov, A., et Kingdom, U. (2007). Openfoam: A c++ library for complex physics simulations. *International Workshop on Coupled Methods in Numerical Dynamics, IUC*, pages 1–20.

Johnsen, S. F., Taylor, Z. A., Clarkson, M. J., Hipwell, J., Modat, M., Eiben, B., Han, L., Hu, Y., Mertzanidou, T., Hawkes, D. J., et Ourselin, S. (2015). NiftySim: A GPU-based nonlinear finite element package for simulation of soft tissue biomechanics. *Int. J. Comput. Assist. Radiol. Surg.*, 10(7):1077–1095.

Kirby, R. C. (2004). Algorithm 839: FIAT, a New Paradigm for Computing Finite Element Basis Functions. *ACM Trans. Math. Softw.*, 30(4):502–516.

Kirby, R. C. et Logg, A. (2006). A Compiler for Variational Forms. *ACM Trans. Math. Softw.*, 32(3):417–444.

Klein, S., Staring, M., Murphy, K., Viergever, M. A., et Pluim, J. P. W. (2010). Elastix: A toolbox for intensity-based medical image registration. *IEEE Trans. Med. Imaging*, 29(1):196–205.

Korelc, J. (2002). Multi-language and multi-environment generation of nonlinear finite element codes. *Engineering with Computers*, 18:312–327.

Korelc, J. (2022). AceGen/AceFEM website and user manuals.

Landreau, P., Drouillard, A., Launoy, G., Ortega-Deballon, P., Jooste, V., Lepage, C., Faivre, J., Facy, O., et Bouvier, A.-M. (2015). Incidence and survival in late liver metastases of colorectal cancer. *J. Gastroenterol. Hepatol.*, 30(1):82–85.

Lange, T., Eulenstein, S., Hünerbein, M., et Schlag, P.-M. (2003). Vessel-based non-rigid registration of MR/CT and 3D ultrasound for navigation in liver surgery. *Comput. Aided Surg.*, 8(5):228–240.

Lasso, A., Heffter, T., Rankin, A., Pinter, C., Ungi, T., et Fichtinger, G. (2014). PLUS: open-source toolkit for ultrasound-guided intervention systems. *IEEE Trans. Biomed. Eng.*, 61(10):2527–2537.

Lavigne, T., Sciumè, G., Laporte, S., Pillet, H., Urcun, S., Wheatley, B., et Rohan, P.-Y. (2022). Société de biomécanique young investigator award 2021: Numerical investigation of the time-dependent stress–strain mechanical behaviour of skeletal muscle tissue in the context of pressure ulcer prevention. *Clinical Biomechanics*, 93:105592.

LeCun, Y. et Bengio, Y. (1998). Convolutional networks for images, speech, and time series. In *The handbook of brain theory and neural networks*, pages 255–258. MIT Press, London, England.

Lengiewicz, J., Habera, M., Zilian, A., et Bordas, S. (2021). Interfacing acegen and fenics for advanced constitutive models. In Baratta, I., Dokken, J. S., Richarson, C., et Scroggs, M. W., editors, *Proceedings of FEniCS 2021, online, 22–26 March*, page 474.

Li, H., Sumner, R. W., et Pauly, M. (2008). Global correspondence optimization for non-rigid registration of depth scans. *Comput. Graph. Forum*, 27(5):1421–1430.

Li, Q., Shen, Z., Li, Q., Barratt, D. C., Dowrick, T., Clarkson, M. J., Vercauteren, T., et Hu, Y. (2022). Trackerless freehand ultrasound with sequence modelling and auxiliary transformation over past and future frames.

Liao, M., Zhang, Q., Wang, H., Yang, R., et Gong, M. (2009). Modeling deformable objects from a single depth camera. In *2009 IEEE 12th International Conference on Computer Vision*, pages 167–174. IEEE.

Llovet, J. M. (2005). Updated treatment approach to hepatocellular carcinoma. *J. Gastroenterol.*, 40(3):225–235.

Luo, M., Yang, X., Wang, H., Du, L., et Ni, D. (2022). Deep motion network for freehand 3D ultrasound reconstruction. In *Lecture Notes in Computer Science*, Lecture notes in computer science, pages 290–299. Springer Nature Switzerland, Cham.

Maas, S., Ellis, B., Ateshian, G., et Weiss, J. (2012). Febio: Finite elements for biomechanics. *Journal of biomechanical engineering*, 134:011005.

Machi, J., Isomoto, H., Yamashita, Y., Kurohiji, T., Shirouzu, K., et Kakegawa, T. (1987). Intraoperative ultrasonography in screening for liver metastases from colorectal cancer: comparative accuracy with traditional procedures. *Surgery*, 101(6):678–684.

Madhok, B., Nanayakkara, K., et Mahawar, K. (2022). Safety considerations in laparoscopic surgery: A narrative review. *World J. Gastrointest. Endosc.*, 14(1):1–16.

Maes, F., Collignon, A., Vandermeulen, D., Marchal, G., et Suetens, P. (1997). Multimodality image registration by maximization of mutual information. *IEEE Trans. Med. Imaging*, 16(2):187–198.

Malgat, R., Gilles, B., Levin, D. I. W., Nesme, M., et Faure, F. (2015). Multifarious hierarchies of mechanical models for artist assigned levels-of-detail. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '15, page 27–36, New York, NY, USA. Association for Computing Machinery.

Martins, P. A. L. S., Natal Jorge, R. M., et Ferreira, A. J. M. (2006). A comparative study of several material models for prediction of hyperelastic properties: Application to silicone-rubber and soft tissues. *Strain*, 42(3):135–147.

Mazier, A., Bilger, A., Forte, A., Peterlik, I., Hale, J., et Bordas, S. (2022). Inverse deformation analysis: an experimental and numerical assessment using the fenics project. *Engineering with Computers*.

Mazier, Arnaud and El Hadramy, Sidaty, Brunet, J.-N., HALE, J., Cotin, S., et BORDAS, S. (August 2022). Sonics: Interfacing sofa and fenics for advanced constitutive models.

Mazier, Arnaud and El Hadramy, Sidaty, Brunet, J.-N., Hale, J. S., Cotin, S., et Bordas, S. P. A. (2024). Sonics: develop intuition on biomechanical systems through interactive error controlled simulations. *Eng. Comput.*, 40(3):1857–1876.

Mehrabian, H. et Samani, A. (2009). Constrained hyperelastic parameters reconstruction of PVA (polyvinyl alcohol) phantom undergoing large deformation. In *Medical Imaging 2009: Visualization, Image-Guided Procedures, and Modeling*. SPIE.

Melzi, S., Ren, J., Rodolà, E., Sharma, A., Wonka, P., et Ovsjanikov, M. (2019). ZoomOut. *ACM Trans. Graph.*, 38(6):1–14.

Mercier, L., Langø, T., Lindseth, F., et Collins, D. L. (2005). A review of calibration techniques for freehand 3-D ultrasound systems. *Ultrasound Med. Biol.*, 31(4):449–471.

Meskell, M. (2010). Principles of anatomy and physiology. *J. Anat.*, 217(5):631–631.

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M. J., Terrel, A. R., Roučka, v., Saboo, A., Fernando, I., Kulal, S., Cimrman, R., et Scopatz, A. (2017). Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103.

Mihai, L. A., Budday, S., Holzapfel, G. A., Kuhl, E., et Goriely, A. (2017). A family of hyperelastic models for human brain tissue. *Journal of the Mechanics and Physics of Solids*, 106:60–79.

Mintz, G. S., Popma, J. J., Pichard, A. D., Kent, K. M., Satler, L. F., Chuang, Y. C., Ditrano, C. J., et Leon, M. B. (1995). Patterns of calcification in coronary artery disease. a statistical analysis of intravascular ultrasound and coronary angiography in 1155 lesions. *Circulation*, 91(7):1959–1965.

Mirnezami, R., Mirnezami, A. H., Chandrakumaran, K., Abu Hilal, M., Pearce, N. W., Primrose, J. N., et Sutcliffe, R. P. (2011). Short- and long-term outcomes after laparoscopic and open hepatic resection: systematic review and meta-analysis. *HPB (Oxford)*, 13(5):295–308.

Miura, K., Ito, K., et Aoki, T. (2021). Jun ohmiya, and satoshi kondo probe localization from ultrasound image sequences using deep learning for volume reconstruction. *International Forum on Medical Imaging in Asia*, 11792.

Mohamed, F. et Vei Siang, C. (2019). A survey on 3D ultrasound reconstruction techniques. In *Artificial Intelligence - Applications in Medicine and Biology*. IntechOpen.

Mohammadi, Z. et Keyvanpour, M. R. (2021). Similarity measures in medical image registration a review article. In *2021 12th International Conference on Information and Knowledge Technology (IKT)*, pages 89–95. IEEE.

Mooney, M. (1940). A theory of large elastic deformation. *Journal of Applied Physics*, 11(9):582–592.

Moré, J. J. (1978). The Levenberg-Marquardt algorithm: Implementation and theory. In *Lecture Notes in Mathematics*, Lecture notes in mathematics, pages 105–116. Springer Berlin Heidelberg, Berlin, Heidelberg.

Mozaffari, M. H. et Lee, W.-S. (2017). Freehand 3-D ultrasound imaging: A systematic review. *Ultrasound Med. Biol.*, 43(10):2099–2124.

Myronenko, A. et Song, X. (2010). Point set registration: coherent point drift. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(12):2262–2275.

Nakamoto, M., Hirayama, H., Sato, Y., Konishi, K., Kakeji, Y., Hashizume, M., et Tamura, S. (2007). Recovery of respiratory motion and deformation of the liver using laparoscopic freehand 3D ultrasound system. *Med. Image Anal.*, 11(5):429–442.

Nam, W. H., Kang, D.-G., Lee, D., Lee, J. Y., et Ra, J. B. (2012). Automatic registration between 3D intra-operative ultrasound and pre-operative CT images of the liver based on robust edge matching. *Phys. Med. Biol.*, 57(1):69–91.

Narayanan, H. (2012). A computational framework for nonlinear elasticity. In Logg, A., Mardal, K.-A., et Wells, G., editors, *Automated Solution of Differential Equations by the Finite Element Method*, number 84 in Lecture Notes in Computational Science and Engineering, pages 525–541. Springer Berlin Heidelberg.

Narkhede, M. V., Bartakke, P. P., et Sutaone, M. S. (2022). A review on weight initialization strategies for neural networks. *Artif. Intell. Rev.*, 55(1):291–322.

Nguyen, V. P., Rabczuk, T., Bordas, S., et Duflot, M. (2008). Meshless methods: A review and computer implementation aspects. *Mathematics and Computers in Simulation*, 79(3):763–813.

Nguyen-Thanh, V. M., Zhuang, X., et Rabczuk, T. (2019). A deep energy method for finite deformation hyperelasticity. *European Journal of Mechanics - A/Solids*, page 103874.

Nikolaev, S. et Cotin, S. (2020). Estimation of boundary conditions for patient-specific liver simulation during augmented surgery. *Int. J. Comput. Assist. Radiol. Surg.*, 15(7):1107–1115.

Ning, G., Liang, H., Zhou, L., Zhang, X., et Liao, H. (2022). Spatial position estimation method for 3D ultrasound reconstruction based on hybrid transfomers. In *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*. IEEE.

Niroomandi, S., Alfaro, I., Cueto, E., et Chinesta, F. (2008). Real-time deformable models of non-linear tissues by model reduction techniques. *Comput. Methods Programs Biomed.*, pages 223–231.

Niroomandi, S., González, D., Alfaro, I., Bordeu, F., Leygue, A., Cueto, E., et Chinesta, F. (2013). Real time simulation of biological soft tissues : A pgd approach. *International journal for numerical methods in biomedical engineering*, 29.

Ogden, R. (1972). Large deformation isotropic elasticity – on the correlation of theory and experiment for incompressible rubberlike solids. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 326(1567):565–584.

Oliveira, F. P. M. et Tavares, J. M. R. S. (2014). Medical image registration: a review. 17(2):73–93.

Orcutt, S. T. et Anaya, D. A. (2018). Liver resection and surgical strategies for management of primary liver cancer. *Cancer Control*, 25(1):1073274817744621.

Ortiz, J. J. G., Guttag, J., et Dalca, A. (2023). Magnitude invariant parametrizations improve hypernetwork learning.

Ovsjanikov, M., Ben-Chen, M., Solomon, J., Butscher, A., et Guibas, L. (2012). Functional maps. *ACM Trans. Graph.*, 31(4):1–11.

Parviz, Y., Shlofmitz, E., Fall, K. N., Konigstein, M., Maehara, A., Jeremias, A., Shlofmitz, R. A., Mintz, G. S., et Ali, Z. A. (2018). Utility of intracoronary imaging in the cardiac catheterization laboratory: comprehensive evaluation with intravascular ultrasound and optical coherence tomography. *Br. Med. Bull.*, 125(1):79–90.

Pascanu, R., Mikolov, T., et Bengio, Y. (2012). On the difficulty of training recurrent neural networks.

Patte, C., Genet, M., et Chapelle, D. (2022). A quasi-static poromechanical model of the lungs. *Biomechanics and Modeling in Mechanobiology*, 21(2):527–551.

Payan, Y. et Ohayon, J. (2017). Preface. In Payan, Y. et Ohayon, J., editors, *Biomechanics of Living Organs*, volume 1 of *Translational Epigenetics*, pages xxv–xxvi. Academic Press, Oxford.

Penney, G., Blackall, J., Hayashi, D., Sabharwal, T., Adam, A. N., et Hawkes, D. (2001). Overview of an ultrasound to CT or MR registration system for use in thermal ablation of liver metastases.

Penney, G. P., Blackall, J. M., Hamady, M. S., Sabharwal, T., Adam, A., et Hawkes, D. J. (2004). Registration of freehand 3D ultrasound and magnetic resonance liver images. *Med. Image Anal.*, 8(1):81–91.

Peterlik, I., Courtecuisse, H., Duriez, C., et Cotin, S. (2014). Model-based identification of anatomical boundary conditions in living tissues. In *Information Processing in Computer-Assisted Interventions*, Lecture notes in computer science, pages 196–205. Springer International Publishing, Cham.

Pezzuto, S., Ambrosi, D., et Quarteroni, A. (2014). An orthotropic active–strain model for the myocardium mechanics and its numerical approximation. *European Journal of Mechanics - A/Solids*, 48:83–96.

Plantefeve, R., Peterlik, I., Haouchine, N., et Cotin, S. (2016). Patient-specific biomechanical modeling for guidance during minimally-invasive hepatic surgery. *Ann. Biomed. Eng.*, 44(1):139–153.

Pohlman, R. M., Turney, M. R., Wu, P.-H., Brace, C. L., Ziemlewicz, T. J., et Varghese, T. (2019). Two-dimensional ultrasound-computed tomography image registration for monitoring percutaneous hepatic intervention. *Med. Phys.*, 46(6):2600–2609.

Porter, B. C., Rubens, D. J., Strang, J. G., Smith, J., Totterman, S., et Parker, K. J. (2001). Three-dimensional registration and fusion of ultrasound and MRI using major vessels as fiducial markers. *IEEE Trans. Med. Imaging*, 20(4):354–359.

Pratt, G., Shin, C., et Hicks (1998). Gauss-Newton and full newton methods in frequency-space seismic waveform inversion. *Geophys. J. Int.*, 133(2):341–362.

Prevost, R., Salehi, M., Sprung, J., Ladikos, A., Bauer, R., et Wein, W. (2017). Erratum to: Deep learning for sensorless 3D freehand ultrasound imaging. In *Lecture Notes in Computer Science*, Lecture notes in computer science, pages E1–E1. Springer International Publishing, Cham.

Raissi, M., Perdikaris, P., et Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378:686–707.

Ralston, A. et Rabinowitz, P. (2001). A first course in numerical analysis (2nd ed.). In *A First Course in Numerical Analysis (Second Edition)*, page i. Dover Publications, New York, second edition edition.

Ramalhinho, J., Robu, M., Thompson, S., Edwards, P., Schneider, C., Gurusamy, K., Hawkes, D., Davidson, B., Barratt, D., et Clarkson, M. J. (2017). Breathing motion compensated registration of laparoscopic liver ultrasound to CT. In Webster, R. J. et Fei, B., editors, *Medical Imaging 2017: Image-Guided Procedures, Robotic Interventions, and Modeling*, volume 10135, pages 735–743. SPIE.

Ramalhinho, J., Tregidgo, H., Allam, M., Travlou, N., Gurusamy, K., Davidson, B., Hawkes, D., Barratt, D., et Clarkson, M. J. (2019). Registration of untracked 2D laparoscopic ultrasound liver images to CT using content-based retrieval and kinematic priors. In *Smart Ultrasound Imaging and Perinatal, Preterm and Paediatric Image Analysis*, Lecture notes in computer science, pages 11–19. Springer International Publishing, Cham.

Ramesh, K. K. D., Kumar, G., Swapna, K., Datta, D., et Rajest, S. (2018). A review of medical image segmentation algorithms. *EAI Endorsed Trans. Pervasive Health Technol.*, page 169184.

Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., et Koltun, V. (2022). Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3).

Rappel, H., Beex, L., Hale, J., Noels, L., et Bordas, S. (2019). A tutorial on bayesian inference to identify material parameters in solid mechanics. *Archives of Computational Methods in Engineering*, 27.

Rathgeber, F., Ham, D. A., Mitchell, L., Lange, M., Luporini, F., Mcrae, A. T. T., Bercea, G.-T., Markall, G. R., et Kelly, P. H. J. (2016). Firedrake: Automating the Finite Element Method by Composing Abstractions. *ACM Transactions on Mathematical Software (TOMS)*, 43(3):24:1–24:27.

Reich, H., McGlynn, F., DeCaprio, J., et Budin, R. (1991). Laparoscopic excision of benign liver lesions. *Obstet. Gynecol.*, 78(5 Pt 2):956–958.

Rivaz, H., Chen, S. J.-S., et Collins, D. L. (2015). Automatic deformable MR-ultrasound registration for image-guided neurosurgery. *IEEE Trans. Med. Imaging*, 34(2):366–380.

Rivlin, R. S. (1948). Large elastic deformations of isotropic materials IV. further developments of the general theory. *Philos. Trans. R. Soc. Lond.*, 241(835):379–397.

Roche, A., Malandain, G., et Ayache, N. (2000). Unifying maximum likelihood approaches in medical image registration. *Int. J. Imaging Syst. Technol.*, 11(1):71–80.

Rodenberg, B., Desai, I., Hertrich, R., Jaust, A., et Uekermann, B. (2021). FEniCS–preCICE: Coupling FEniCS to other simulation software. *SoftwareX*, 16:100807.

Roewer-Despres, F., Khan, N., et Stavness, I. (2018). Towards finite-element simulation using deep learning.

Ronneberger, O., Fischer, P., et Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *Lecture Notes in Computer Science*, Lecture notes in computer science, pages 234–241. Springer International Publishing, Cham.

Rosenfeld, A. (1976). *Digital picture processing*. Academic press.

S., E. H., J., V., N., P., et S., C. (2024a). Towards real-time vessel guided augmented reality for liver surgery. In *2024 IEEE International Symposium on Biomedical Imaging (ISBI)*, volume 1, pages 1–5. IEEE.

S., E. H., N., P., et S., C. (2024b). HyperU-Mesh: Real-time deformation of soft-tissues across variable patient-specific parameters. In *Computational Biomechanics Workshop at MICCAI 2024*.

Sarker, I. H. (2021). Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. *SN Comput. Sci.*, 2(6):420.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., et Monfardini, G. (2009). The graph neural network model. *IEEE Trans. Neural Netw.*, 20(1):61–80.

Schneider, J. et Vlachos, M. (2023). A survey of deep learning: From activations to transformers. In *International Conference on Agents and Artificial Intelligence*.

Schullian, P., Johnston, E. W., Putzer, D., Eberle, G., Laimer, G., et Bale, R. (2020). Safety and efficacy of stereotactic radiofrequency ablation for very large (≥8 cm) primary and metastatic liver tumors. *Sci. Rep.*, 10(1):1618.

Scroggs, M. W., Baratta, I. A., Richardson, C. N., , et Wells, G. N. (2022). Basix: a runtime finite element basis evaluation library. submitted to Journal of Open Source Software.

Seo, M., Kim, D., Lee, K., Hong, S., Bae, J. S., Hoon Kim, J., et Kwak, S. (2021). Neural contrast enhancement of CT image. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 3972–3981. IEEE.

Servin, F., Collins, J. A., Heiselman, J. S., Frederick-Dyer, K. C., Planz, V. B., Geevarghese, S. K., Brown, D. B., Jarnagin, W. R., et Miga, M. I. (2024). Simulation of image-guided microwave ablation therapy using a digital twin computational model. *IEEE Open J. Eng. Med. Biol.*

Shahin, O., Beširević, A., Kleemann, M., et Schlaefer, A. (2014). Ultrasound-based tumor movement compensation during navigated laparoscopic liver interventions. *Surg. Endosc.*, 28(5):1734–1741.

Shekhar, R., Dandekar, O., Bhat, V., Philip, M., Lei, P., Godinez, C., Sutton, E., George, I., Kavic, S., Mezrich, R., et Park, A. (2010). Live augmented reality: a new visualization method for laparoscopic surgery using continuous volumetric computed tomography. *Surg. Endosc.*, 24(8):1976–1985.

Shi, J. et Tomasi (1994). Good features to track. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94*. IEEE Comput. Soc. Press.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., et Woo, W.-C. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. pages 802–810.

Simon, B. R. (1992). Multiphase Poroelastic Finite Element Models for Soft Tissue Structures. *Applied Mechanics Reviews*, 45(6):191–218.

Sinaie, S., Nguyen, V. P., Thanh Nguyen, C., et Bordas, S. (2017). Programming the material point method in julia. *Advances in Engineering Software*, 105.

Smith, M. (2009). *ABAQUS/Standard User's Manual, Version 6.9*. Dassault Systèmes Simulia Corp, United States.

Sokooti, H., de Vos, B., Berendsen, F., Lelieveldt, B. P. F., Išgum, I., et Staring, M. (2017). Nonrigid image registration using multi-scale 3D convolutional neural networks. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2017*, Lecture notes in computer science, pages 232–239. Springer International Publishing, Cham.

Soler, L., Hostettler, A., Agnus, V., Charnoz, A., Fasquel, J., Moreau, J., Osswald, A., Bouhadjar, M., et Marescaux, J. (2010). 3d image reconstruction for comparison of algorithm database: A patient specific anatomical and medical image database. *Tech. Rep, vol. 1, no 1*.

Song, Y., Totz, J., Thompson, S., Johnsen, S., Barratt, D., Schneider, C., Gurusamy, K., Davidson, B., Ourselin, S., Hawkes, D., et Clarkson, M. J. (2015). Locally rigid, vessel-based registration for laparoscopic liver surgery. *Int. J. Comput. Assist. Radiol. Surg.*, 10(12):1951–1961.

Sonka, M., Zhang, X., Siebes, M., Bissing, M. S., Dejong, S. C., Collins, S. M., et McKay, C. R. (1995). Segmentation of intravascular ultrasound images: a knowledge-based approach. *IEEE Trans. Med. Imaging*, 14(4):719–732.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., et Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Steele, Jr, G. et Ravikumar, T. S. (1989). Resection of hepatic metastases from colorectal cancer biologicperspectives. *Ann. Surg.*, 210(2):127–138.

Studholme, C., Hill, D. L., et Hawkes, D. J. (1997). Automated three-dimensional registration of magnetic resonance and positron emission tomography brain images by multiresolution optimization of voxel similarity measures. *Med. Phys.*, 24(1):25–35.

Studholme, C., Hill, D. L. G., et Hawkes, D. J. (1999). An overlap invariant entropy measure of 3D medical image alignment. *Pattern Recognit.*, 32(1):71–86.

Suganyadevi, S., Seethalakshmi, V., et Balasamy, K. (2022). A review on deep learning in medical image analysis. *Int. J. Multimed. Inf. Retr.*, 11(1):19–38.

Sumner, R. W., Schmid, J., et Pauly, M. (2007). Embedded deformation for shape manipulation. *ACM Trans. Graph.*, 26(3):80.

Suwelack, S., Röhl, S., Bodenstedt, S., Reichard, D., Dillmann, R., dos Santos, T., Maier-Hein, L., Wagner, M., Wünscher, J., Kenngott, H., Müller, B. P., et Speidel, S. (2014). Physics-based shape matching for intraoperative image guidance. *Med. Phys.*, 41(11):111901.

Tagliabue, E., Piccinelli, M., Dall'Alba, D., Verde, J., Pfeiffer, M., Marin, R., Speidel, S., Fiorini, P., et Cotin, S. (2021). Intra-operative update of boundary conditions for patient-specific surgical simulation. *Medical Image Computing and Computer Assisted Intervention – MICCAI 2021*, pages 373–382.

Talebi, H., Silani, M., Bordas, S. P. A., Kerfriden, P., et Rabczuk, T. (2013). A computational library for multiscale modeling of material failure. *Computational Mechanics*, 53(5):1047–1071.

Terven, J., Cordova-Esparza, D. M., Ramirez-Pedraza, A., Chavez-Urbiola, E. A., et Romero-Gonzalez, J. A. (2023). Loss functions and metrics in deep learning.

Tingting, X. et Ning, W. (2013). Non-rigid multi-modal medical image registration: A review. In *Proceedings of 3rd International Conference on Multimedia Technology(ICMT-13)*, Paris, France. Atlantis Press.

Tonutti, M., Gras, G., et Yang, G.-Z. (2017). A machine learning approach for real-time modelling of tissue deformation in image-guided neurosurgery. *Artif. Intell. Med.*, 80:39–47.

Urade, T., Verde, J. M., García Vázquez, A., Gunzert, K., Pessaux, P., Marescaux, J., et Giménez, M. E. (2021). Fluoroless intravascular ultrasound image-guided liver navigation in porcine models. *BMC Gastroenterol.*, 21(1):24.

Urcun, S., Rohan, P.-Y., Sciumè, G., et Bordas, S. (2021a). Cortex tissue relaxation and slow to medium load rates dependency can be captured by a two-phase flow poroelastic model. *Journal of the Mechanical Behavior of Biomedical Materials*, 126:104952.

Urcun, S., Rohan, P.-Y., Skalli, W., Nassoy, P., Bordas, S. P. A., et Sciumè, G. (2021b). Digital twinning of cellular capsule technology: Emerging outcomes from the perspective of porous media mechanics. *PLOS ONE*, 16(7):1–30.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., et Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., et Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Veronda, D. et Westmann, R. (1970). Mechanical characterization of skin—finite deformations. *Journal of Biomechanics*, 3(1):111–124.

Viola, P. et Wells, III, W. M. (1997). *Int. J. Comput. Vis.*, 24(2):137–154.

Wang, Z.-Y., Chen, Q.-L., Sun, L.-L., He, S.-P., Luo, X.-F., Huang, L.-S., Huang, J.-H., Xiong, C.-M., et Zhong, C. (2019). Laparoscopic versus open major liver resection for hepatocellular carcinoma: systematic review and meta-analysis of comparative cohort studies. *BMC Cancer*, 19(1):1047.

Wasserthal, J., Breit, H.-C., Meyer, M. T., Pradella, M., Hinck, D., Sauter, A. W., Heye, T., Boll, D. T., Cyriac, J., Yang, S., Bach, M., et Segeroth, M. (2023). TotalSegmentator: Robust segmentation of 104 anatomic structures in CT images. *Radiol. Artif. Intell.*, 5(5):e230024.

Wein, W., Brunke, S., Khamene, A., Callstrom, M. R., et Navab, N. (2008). Automatic CT-ultrasound registration for diagnostic imaging and image-guided intervention. *Med. Image Anal.*, 12(5):577–585.

Wein, W., Khamene, A., Clevert, D.-A., Kutter, O., et Navab, N. (2007). Simulation and fully automatic multimodal registration of medical ultrasound. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2007*, Lecture notes in computer science, pages 136–143. Springer Berlin Heidelberg, Berlin, Heidelberg.

Wells, 3rd, W. M., Viola, P., Atsumi, H., Nakajima, S., et Kikinis, R. (1996). Multi-modal volume registration by maximization of mutual information. *Med. Image Anal.*, 1(1):35–51.

Woods, R. P., Cherry, S. R., et Mazziotta, J. C. (1992). Rapid automated algorithm for aligning and reslicing PET images. *J. Comput. Assist. Tomogr.*, 16(4):620–633.

Wriggers, P. (2008). *Nonlinear Finite Element Methods*. Springer, Berlin, Germany, 2008 edition.

Wu, J., Westermann, R., et Dick, C. (2014). Real-time haptic cutting of high-resolution soft tissues. *Studies in health technology and informatics*, 196:469–75.

Xie, Y., Liao, H., Zhang, D., Zhou, L., et Chen, F. (2021). Image-based 3D ultrasound reconstruction with optical flow via pyramid warping network. In *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE.

Xu, L., Liu, J., Zhan, W., et Gu, L. (2013). A novel algorithm for CT-ultrasound registration. In *2013 IEEE Point-of-Care Healthcare Technologies (PHT)*. IEEE.

Yan, K., Wang, X., Lu, L., et Summers, R. M. (2018). DeepLesion: automated mining of large-scale lesion annotations and universal lesion detection with deep learning. *J. Med. Imaging (Bellingham)*, 5(03):1.

Yang, X., Kwitt, R., Styner, M., et Niethammer, M. (2017). Quicksilver: Fast predictive image registration – a deep learning approach. *Neuroimage*, 158:378–396.

Zeraatpisheh, M., Bordas, S. P., et Beex, L. A. (2021). Bayesian model uncertainty quantification for hyperelastic soft tissue models. *Data-Centric Engineering*, 2:e9.

Zhang, H. (2021). Image registration. In *Advances in Magnetic Resonance Technology and Applications*, pages 83–94. Elsevier.

Zhou, J. et Fung, Y. C. (1997). The degree of nonlinearity and anisotropy of blood vessel elasticity. *Proceedings of the National Academy of Sciences*, 94(26):14255–14260.

Zienkiewicz, O., Taylor, R., et Fox, D. (2014). The finite element method for solid and structural mechanics. In *The Finite Element Method for Solid and Structural Mechanics (Seventh Edition)*, page i. Butterworth-Heinemann, Oxford, seventh edition edition.

Ølgaard, K. B. et Wells, G. N. (2012). Applications in solid mechanics. In Logg, A., Mardal, K.-A., et Wells, G., editors, *Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book*, Lecture Notes in Computational Science and Engineering, pages 505–524. Springer, Berlin, Heidelberg.

List of Figures

LIST OF TABLES