Machine Learning meets real-time Numerical Simulation - Application to surgical training, preoperative planning and surgical assistance

Andrea Mendizabal

UNIVERSITÉ DE STRASBOURG

ÉCOLE DOCTORALE MATHÉMATIQUES,

SCIENCES DE L'INFORMATION ET DE L'INGÉNIEUR

THÈSE présentée par:

Andrea Mendizabal

soutenue le : **3 Décembre 2020**

pour obtenir le grade de: **Docteur de l'université de Strasbourg**

Discipline/ Spécialité: Informatique et Mathématique

Combinaison entre simulation numérique et apprentissage automatique

Applications à la formation, la planification préopératoire et l'assistance peropératoire

THÈSE dirigée par :

Monsieur COTIN Stéphane	HDR, Université de Strasbourg

RAPPORTEURS :

Monsieur CUETO Elias	HDR, Universidad de Zaragoza
Monsieur MARTIN-GUERRERO José David	Professeur, Universidad de Valencia

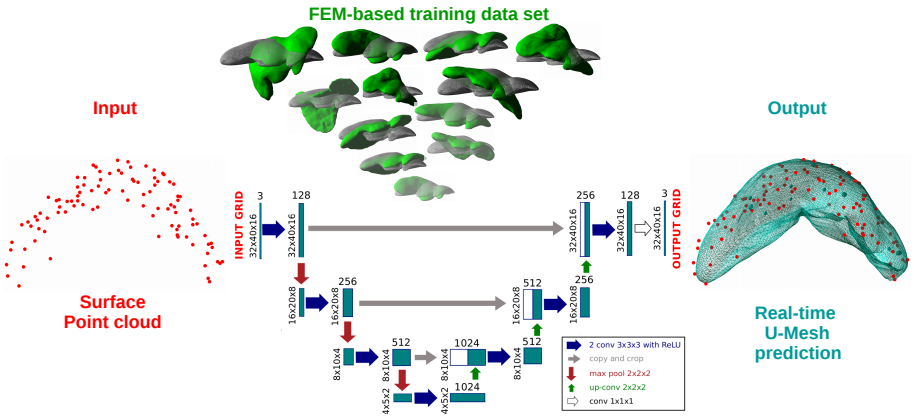
---

AUTRES MEMBRES DU JURY :

Monsieur PAYAN Yohan	HDR, CNRS Laboratoire TIMC-IMAG
Monsieur CHAKRABORTY Souvik	MCF, Indian Institute of Technology Delhi

Machine Learning meets real-time Numerical Simulation

Application to surgical training, preoperative planning and surgical assistance



Andrea Mendizabal

Mathématiques, Sciences de l'Information et de l'Ingénieur (MSII)

Laboratoire des sciences de l'ingénieur, de l'informatique et de l'imagerie (ICube)

University of Strasbourg

Submitted in partial satisfaction of the requirements for the
Degree of Doctor of Philosophy
in Computer Science

Supervisor Stéphane Cotin

Defended on the 3rd of December 2020

Acknowledgements

"*They did not know it was impossible so they did it*" quote by Mark Twain.

This work is the result of three intense years within which I have evolved and grown as a scientist and as a person. The first year was extremely difficult in terms of motivation and also uncertain. But thanks to the support of the persons surrounding me, the realisation of this work has been possible.

I would first like to thank my advisor Stéphane Cotin without who I would still be lost somewhere in the Amazonian forest in Peru. Thank you for your humanity and understanding my concerns. Thank you for your time and dedication, and for sharing and spreading your passion for research!

I would like to express my gratitude to Elias Cueto, to José David Martín-Guerrero, to Yohan Payan and to Souvik Chakraborty for accepting being members of the jury of my PhD defense. I would like to particularly thank José David Martín-Guerrero's team and María José Ruperez Moreno for the inspiring scientific conversations and for being a reference starting point of my PhD.

I would also like to thank Raphael Sznitman and the ARTORG center in Bern, for the support in publishing my first PhD paper. Thank you Pablo Marquez-Neila for your precious help in the technical aspects of deep learning. Thank you Tatiana Fountoukidou and Jan Hermann for the patience when running the experiments with the OCT and the robot arm.

I feel grateful for the amazing time spent in the Mimesis team: you guys are awesome! Thank you for the inspiring conversations and for the sacred breaks playing Quoinche! I would like to particularly thank my deskmate Jean-Nicolas Brunet for all the precious advice, the patience when I did not understand, for being a real friend and listening when I needed to be heard. Thank you Antoine Petit and Sergei Nikolaev for not letting me alone at Clovis during the paper deadlines. Thank you Alban Odot for turning my research scripts into a usable optimized plugin and also for your bad jokes ;) Thank you Hugo Talbot and Rémi Bessard for the rooftop garden! And big thank you to all the team members for the smiles, the croissants, the expertise!

I also would like to thank very important persons in my life whose support was essential for the realization of this work. Gracias Aaron por haber estado a mi lado

a lo largo de este viaje, de principio a fín. Gracias Amoñi por estar cerca mío, ahora y siempre. Merci Gérard de m'avoir rappelé que je suis humaine et pas une machine. Sans toi, j'aurais brûlé toutes mes cartouches à mi-chemin. Thank you Lou for showing me the importance of listening to my body and to my intuition. Thank you Nibiru and Nébula for supporting my mental health, in particular during the lock-down. Thank you to all of my friends that loved me despite my bad humor and unavailability. In particular thank you Adèle and Gaby for being there. Gracias ama, aita y Mario, por creer y confiar en mí, y por haberme dado el tiempo y el espacio necesarios.

I would also like to thank my self for all the effort, the confidence and perseverance dedicated to this work.

Publications

1. Mendizabal, A., Fountoukidou, T., Hermann, J., Sznitman, R., Cotin, S., 2018. A Combined Simulation and Machine Learning Approach for Image-Based Force Classification During Robotized Intravitreal Injections. In International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI 2018), pp. 12-20.

2. Mendizabal, A., Sznitman, R., Cotin, S., 2019. Force classification during robotic interventions through simulation-trained neural networks. International journal of computer assisted radiology and surgery (IJCARS 2019), 14(9), pp. 1601-1610.

3. Mendizabal, A., Márquez-Neila, P., and Cotin, S., 2020. Simulation of hyperelastic materials in real-time using deep learning. Journal of Medical Image Analyses. Volume 59, 101569 10.1016/j.media.2019.101569.

4. Mendizabal, A., Brunet, J.N., Petit, A., Golse, N., Vibert, E., Cotin, S., 2019. Physics-based deep neural network for augmented reality during liver surgery. International Conference on Medical image computing and computer-assisted intervention (MICCAI 2019), pp. 137-145.

5. Mendizabal, A., Tagliabue, E., Brunet, J-N., Dall'Alba,D., Fiorini, P., Cotin, S., 2019. Physics-based Deep Neural Network for Real-Time Lesion Tracking in Ultrasound-guided Breast Biopsy. Computational Biomechanics for Medicine Workshop.

6. Mendizabal, A., Tagliabue, E., Hoellinger, T., Brunet, J.N., Nikolaev, S., Cotin, S., 2020. Data-driven simulation for augmented surgery. Journal of Advanced Structured Materials.

Notations and abbreviations

General notations

- $< x, y >$: dot product of x and y
- ∇f: gradient of f
- $\frac{\partial f}{\partial x}$: partial derivative of f with respect to x
- $\mathbb{I}$: identity matrix in $\mathbb{R}^{3\times3}$
- $tr(\mathbf{A})$: trace of $\mathbf{A}$
- $div(f)$: divergence of f
- $\dot{u}$: first derivative of u
- ü: second derivative of u
- $p \sim \mathcal{N}(\mu, \sigma)$: variable p follows a normal distribution of mean μ and standard deviation σ

Chapter 2

- f_{ext}: external forces
- f_{int}: internal forces
- m: body mass
- $\mathbf{a}$: acceleration
- ρ: mass density
- $\mathbf{v}$: velocity
- u: displacement field
- Ω: computational domain
- Γ: boundary of Ω
- Γ_D: subpart of Γ with Dirichlet boundary conditions

- Γ_N: subpart of Γ with Neumann boundary conditions
- $\mathbf{p}$: initial position of a particle in Ω
- $\bar{\mathbf{p}}$: deformed position of $\mathbf{p}$
- ϕ: transformation undergone by the solid
- $\mathbb{F}$: deformation gradient in $\mathbb{R}^{3\times3}$
- ϵ: strain tensor
- ϵ_c: right Cauchy-Green strain tensor
- ϵ_g: Green-Lagrange strain tensor
- $\mathbf{n}$: surface normal
- $\mathbf{t}_n$: stress vector linked to normal $\mathbf{n}$
- dS: infinitesimal surface
- dV: infinitesimal volume
- V^e: volume of element e
- σ: stress tensor
- σ_c: Cauchy stress tensor
- σ_p: first Piola-Kirchhoff stress tensor
- σ_s: second Piola-Kirchhoff stress tensor
- $\mathbf{C}$: constitutive matrix (stress-strain matrix)
- $W(\epsilon)$: strain energy function
- λ and μ: Lamé coefficients
- ρ: density of the material
- f^s: surface forces
- f^v: volume forces
- E: Young's modulus
- ν: Poisson's ratio
- N_i: shape function at node i
- $\mathbf{K}$: global stiffness matrix
- $\dot{\mathbf{K}}$: tangent stiffness matrix in the Newton-Raphson solver

- $\mathbf{r}$: internal elastic force vector
- N^e: number of elements in the *finite-element* mesh

Chapter 3

- f: neural network function
- $x = (x_1, ..., x_d)$: input of f
- y: output of f
- θ: parameters of the network
- h_j: artificial neuron j
- $w_j = (w_{j,1}, ..., w_{j,d})$: weights linked to neuron j
- b_j: bias of neuron j
- ϕ: activation function
- L: number of hidden layers
- k: layer index
- $z^{(k)}(x)$: output of a neuron in layer k before activation
- $a^{(k)}(x)$: output of a neuron in layer k after activation
- $W^{(k)}$: weights matrix at layer k (number of rows equal to the number of neurons in layer k and number of columns equal to number of neurons in layer $k-1$
- $b^{(k)}$: weights vector at layer k
- ψ: last layer activation function
- K: convolution kernel
- $\mathcal{L}$: loss function
- α: learning rate
- m: batch size
- N: number of samples in training data set

Chapters 4, 7, 8 and 9

- h: network function
- $n_x \times n_y \times n_z$: resolution of the grid
- $\mathbf{C}$: input constraints (expressed in the grid nodes)

- $\mathbf{U_s}$: input surface displacement (expressed at grid nodes)
- $\mathbf{U_v}$: output volumetric displacement field (expressed at grid nodes)
- N: number of samples in training data set
- M: number of samples in testing data set
- n: number of degrees of freedom of the FE mesh (e.g. number of nodes in sparse grid)
- c: number of channels in the first layer of the U-Net
- k: number of steps in the U-Net
- Γ_N: surface area on which forces are applied (its location varies such that the boundary of the computation domain is completely covered)
- Λ: number of samples per Γ_N
- e: mean norm error of a sample (mean euclidian error in Chapter 8)
- $\bar{e}$: average error over the testing data set
- $\sigma(e)$: standard deviation of the error over the testing data set

Chapter 5

- p: sought parameter
- μ_0 and σ_0: initial mean value and standard deviation of p
- n: number of nodes
- k: number of stiffness parameters
- m: number of elements in the stochastic state vector

Chapter 6

- $\mathbf{f_p}$: external force due to intraocular pressure
- $\mathbf{f_n}$: needle pushing on the surface of the sclera
- $\mathbf{f_g}$: gravitational force
- S: surface area of the eye
- P: intraocular pressure
- $(\mathbf{f_n}, I)$: sample of the data set. $\mathbf{f_n}$ is the input force and I is the corresponding OCT image

Abbreviations

- ANN: artificial neural network
- AR: augmented reality
- BC: boundary condition
- CAI: computer-assisted interventions
- CAS: computer-aided surgery
- CNN: convolutional neural network
- CT: computerized tomography
- FC: fully-connected
- FE: finite-element
- FEM: finite-element method
- FSS: feature space size
- FUS: freehand ultrasound system
- GPU: graphics processing unit
- H8: 8-node hexahedral elements
- HPOD: hyperreduced proper orthogonal decomposition
- IBM: immersed-boundary method
- ICP: iterative closest point
- IOP: intraocular pressure
- MIS: minimally-invasive surgery
- ML: machine learning
- MLP: multi-layer perceptron
- MRI: magnetic resonance imaging
- NN: neural network
- OCT: optical coherence tomography
- PBD: position based dynamics
- PCA: principal-component analysis
- PDF: probability density function

- PGD: proper generalized decomposition
- POD: proper orthogonal decomposition
- ReLU: rectified linear unit activation function
- ROUKF: reduced-order unscented Kalman filter
- SGD: stochastic gradient descent
- T4: 4-node tetrahedral elements
- TRE: target registration error
- US: ultrasound
- VR: virtual reality

Table of Contents

1 General Introduction **2**

I Real-time numerical simulations **8**

2 Biomechanical Problem Formulation **10**

2.1 Elasticity equations for soft tissue simulation 10

2.1.1 A measure of deformation: the strain tensors 10

2.1.2 A measure of internal forces: the stress tensor 12

2.1.3 Constitutive law: linking stress and strain 14

2.1.4 Forces and weak formulation 15

2.2 The Finite-Element Method 17

2.2.1 Domain discretization 17

2.2.2 Shape functions 18

2.2.3 Build the system matrices 19

2.2.4 Solving the system of equations 20

2.3 Model parameterization 21

2.3.1 Material properties 21

2.3.2 Boundary conditions 22

2.4 Conclusion 22

3 Artificial Neural Networks: Fundamentals **24**

3.1 Components of an artificial neural network 24

3.1.1 Definition 24

3.1.2 Artificial neuron and activation functions 24

3.2 Specific types of neural networks 27

3.2.1 Multi-layer perceptron 27

3.2.2 Convolutional neural networks 27

3.3 Training a neural network 29

3.3.1 Loss functions . . . 30
3.3.2 Minimization with Stochastic gradient descent . . . 31
3.3.3 Backpropagation . . . 32
3.4 Conclusion . . . 34

4 The U-Mesh Framework **37**
4.1 Introduction and related work . . . 37
4.2 Displacement field estimation with deep neural networks . . . 40
4.2.1 Network architecture . . . 40
4.2.2 Offline training data generation . . . 43
4.3 Results on synthetic objects . . . 44
4.3.1 Validation metrics . . . 44
4.3.2 U-Mesh applied to a cantilever beam . . . 45
4.3.3 U-Mesh applied to an L-shaped object . . . 48
4.3.4 Comparison of U-Mesh and POD . . . 49
4.4 Conclusion . . . 51

II Patient-specific real-time simulations **55**

5 Patient-specific modeling **57**
5.1 Introduction . . . 57
5.2 Patient-specific geometry . . . 58
5.3 Model parameterization . . . 58
5.3.1 ROUKF: Overview of the algorithm . . . 59
5.4 Conclusion . . . 61

6 Force Estimation in Robotized Injections **63**
6.1 Introduction and related work . . . 63
6.2 Method . . . 65
6.2.1 Training data generation . . . 65
6.2.2 Neural network for image classification . . . 68
6.3 Experimental results . . . 69
6.3.1 Experimental set up . . . 70
6.3.2 Identification of the Young's modulus . . . 71
6.3.3 Training data generation . . . 74
6.3.4 Validation on unseen real data . . . 74

6.4 Conclusion . . . 76

7 The U-Mesh for Augmented Hepatic Surgery 80
7.1 Introduction . . . 80
7.2 Displacement field on regular grids . . . 81
7.3 The U-Mesh applied to a liver . . . 83
7.3.1 Results on a synthetic liver . . . 83
7.3.2 Results in augmented surgery . . . 86
7.4 Conclusion . . . 88

8 The U-Mesh for Ultrasound-guided Breast Biopsy 93
8.1 Introduction and related work . . . 93
8.2 Method . . . 95
8.2.1 Simulation of breast tissue . . . 95
8.2.2 Training data generation . . . 96
8.3 Experiments and Results . . . 97
8.3.1 Predictions on synthetic data . . . 98
8.3.2 Predictions on phantom data . . . 99
8.4 Conclusion . . . 101

9 Towards Patient-Specific Networks 104
9.1 Introduction . . . 104
9.2 Stochastic identification of patient-specific properties . . . 105
9.2.1 Elastic modulus . . . 106
9.2.2 Boundary conditions . . . 107
9.3 Patient-specificity as input to the network . . . 108
9.3.1 Geometry and heterogeneity . . . 109
9.3.2 Material parameters and boundary conditions . . . 110
9.4 Transfer learning for patient-specific simulations . . . 110
9.4.1 Validation metrics . . . 111
9.4.2 Beam with hidden boundary conditions . . . 112
9.4.3 New boundary conditions and sparse data . . . 114
9.5 Conclusion . . . 115

10 General conclusion 118

III Brief summary in french **123**

10.1 Introduction 124
10.2 Estimation de force avec des réseaux de neurones 126
10.3 Plateforme U-Mesh: estimation du champ de déplacement avec des réseaux de neurones 128
10.4 U-Mesh pour la chirurgie augmentée du foie 130
10.5 U-Mesh pour la biopsie du sein 132
10.6 Conclusion 134

Chapter 1
General Introduction

There are many applications in engineering where the deformation of nonlinear structures needs to be simulated in real time, or would benefit from being computed interactively. Some important examples can be found in the field of medicine, in order to develop training systems for learning surgical skills [Ayache et al., 2006] or in the field of surgical navigation, where augmented reality combined with interactive simulations can bring significant improvements to clinical practice [Haouchine et al., 2013a]. Medical robotics, involving flexible robots or interactions with soft tissues, is another important area where real-time simulation of flexible structures is essential, in order to have a better control of the robot [Duriez, 2013].

In the context of *minimally-invasive surgery* (MIS) or laparoscopic surgery for instance, extensive surgical training is mandatory to support the success of the intervention [Rosser et al., 2000]. During this type of surgery, usually three to four small incisions are done in the abdomen, through which surgical tools and a camera are inserted in the abdominal cavity. The abdomen is then inflated with gas, thus creating a working space for the surgeons to operate guided by the images from the camera, displayed on a screen. MIS offers several benefits to the patient such as reducing the bleeding and the risk of infection as well as shortening recovery time [Li et al., 2017]. However, the reduced field of view of the endoscopic camera and the lack of tactile information, make it difficult to succeed in such interventions [Plantefeve et al., 2016]. Surgeons need a lot of dexterity so an extensive training is required. Instead of practicing on animals or on cadavers, virtual laparoscopic training allows surgeons to practice on a completely virtual environment. Through virtual reality and simulation, it is possible to reproduce with high fidelity what the surgeon experiments during the surgery. Apart from the deformation of the organs, the interactions between surgical tools and the tissues are also modeled using haptic devices to mimic tactile sensation. To obtain a stable haptic feedback, simulations must run at 500 frames per second (FPS) [Courtecuisse et al., 2015]. Interventional radiology is another example of intervention requiring specific surgeon training due to its difficulty [Messina et al., 2002]. Indeed, the anatomy is visualized through X-ray images, and the surgical gesture is performed with flexible instruments (such as

catheters) through the vascular tree towards the target. In such training systems, the surgical tools are deformable objects and the vessels are generally considered to be rigid. Such models need to be interactive, therefore the simulations must be computed in real-time (e.g. 30 FPS for visual interaction [Duriez, 2013]).

Computer-assisted interventions (CAI) also require the computation of nonlinear deformations in real-time. For example the use of surgical robots is revolutionizing the field of MIS as they can overcome several human limitations [Troccaz, 2012]. The control loop of the robot is often based on intraoperative images and it must take into account the interactions of the robot with soft tissues, as well as its own deformation in the case of soft robotics. In needle insertion procedures for instance, both the needle and the tissue usually deform as the insertion proceeds and authors in [Adagolodjo et al., 2016] proposed to control the robot through real-time numerical simulations.

On another hand, in the context of hepatic surgery, the objective is to accurately locate the internal structures such as tumors and blood vessels (that need to be preserved for the post-operative regeneration of the liver tissue). While the initial position of these structures is known from the preoperative images, their actual position during surgery is often hidden or uncertain (even for well-trained and experienced surgeons). To help the surgeon overcome these difficulties, augmented reality (AR) techniques are used to enrich visual information through a fusion of intraoperative images and a preoperative 3D model of the patient's anatomy. However, correctly aligning preoperative data to intraoperative images remains an unsolved problem, especially when large deformations are involved and only sparse input data is available. This is typically the case when trying to provide an augmented view of an organ during surgery. In most cases, just about 30% of the surface of the organ is visible due to the limited field of view of the laparoscopic camera or size of the incision [Plantefeve et al., 2016].

In all the cited examples, the considered structures are deformable and can be very soft. Surgical manipulations cause the organs to deform and interact with the surrounding anatomy. Several works have demonstrated the benefits of physics-based models, particularly patient-specific biomechanical models, for accurate registration between different preoperative 3D anatomical model and intraoperative data [Clements et al., 2008, Haouchine et al., 2013b, Suwelack et al., 2014, Alvarez et al., 2018]. While there are different numerical strategies for solving these models, we only consider the finite-element (FE) method throughout this manuscript, for its accuracy and ability to simulate a large range of materials on potentially complex domains. It gives a numerical approximation of a partial derivatives equation discretizing the considered object using nodes connected by elements. The choice of the constitutive model, and its parameterization, are obviously key to an accurate

registration. It is usually acknowledged that a registration of internal structures below $5\,mm$ is needed for best clinical impact, such as targeting relatively small tumors [Ruiter et al., 2006]. Achieving the demanded accuracy within real-time constraints raises several challenges that are far from being solved.

When considering augmented reality or, more generally, real-time tracking of the organ deformation to provide up-to-date guidance, the computational efficiency of the FE method is essential. Indeed, if the simulations are not up-to-date, they might simply be ignored by the clinicians. In the case of augmented hepatic surgery for instance, intraoperative images are acquired at about $20\,Hz$ leading to update times of less than $50\,ms$. During this small amount of time, acquisition and processing of the images as well as model update need to take place. As a result, FE computation times should require less than $30\,ms$. If only small deformations take place, achieving such computation times is feasible [Meier et al., 2005]. However, if large nonlinear deformations happen, computation times become incompatible with such constraints. A solution might be the use of the co-rotational FE method, where geometrical nonlinearities can be handled in real-time [Haouchine et al., 2013a, Petit et al., 2018]. Nevertheless, when more complex biomechanical models need to be used these optimizations no longer hold. Alternative solutions have been proposed with different trade-offs regarding the ratio between computation time and model accuracy [Peterlík et al., 2012, Suwelack et al., 2014, Modrzejewski et al., 2018, Niroomandi et al., 2008, Johnsen et al., 2015, Allard et al., 2007]. [Marchessau et al., 2010] proposed the Multiplicative Jacobian Energy Decomposition (MJED) that allows for fast and realistic liver deformations including hyperelasticity, porosity and viscosity. Also, [Miller et al., 2007] introduced Total Lagrangian explicit dynamics (TLED) which can achieve real-time performances when coupled with explicit time integration and GPU-based solvers [Joldes et al., 2010].

In recent years, machine learning (ML) started to revolutionize several fields (vision, language processing, image recognition, genomics). With sufficient ground truth data, machine learning algorithms can map the input of a function to its output without any mathematical formulation of the problem. The high inferring speed of these methods can be useful for many applications where the prediction speed is of critical importance. Due to this characteristic and the fact that they are driven directly by data, these methods seem promising for the learning of the entire mechanical behavior of the anatomy without relying on prior models.

Some first attempts that exploit learning methods to estimate the deformation of biological tissues have recently been made. By implicitly encoding soft tissue mechanical behavior in the trained ML models, they proved successful to predict the entire 3D organ deformation starting either by applied surface forces

[Morooka et al., 2008, Tonutti et al., 2017, Rechowicz et al., 2013] or by surface displacements [Lorente et al., 2017]. The accuracy of a ML model highly depends on the network architecture and on the quality and on the amount of data used to train it. In an ideal scenario, such a model would be trained with an infinite amount of real patient-specific noise-free data, which is in practice not possible. Within this framework, FE simulations can be exploited to generate synthetic data that is highly representative of the reality, to be used as training samples.

Among the various ML techniques, the use of neural networks (NN) has considerably increased. This is due to the fact that they are the building blocks of deep learning, a class of methods which is able to learn data representations and has demonstrated strong abilities at extracting high-level representations of complex processes. For example, NN are used by [Tonutti et al., 2017] and [Rechowicz et al., 2013] to predict the displacement of brain tumors and of the rib cage surface respectively, starting from the acting forces. However, both these works do not predict whole volume deformation but only surface displacements. NN based methods have been also used to predict liver deformation in augmented surgery. [Morooka et al., 2008] trained a NN to predict liver deformations for a given input force. They use their model together with principal-component analysis to compress the size of the output deformation modes, and thus reduce the training time. Although the model proved able to learn the deformation modes, it was only applied to simulated data and not to real cases. From all these works it emerges that the main advantage of using neural networks to predict anatomical deformations is that the prediction speed is in the order of few milliseconds and is not affected by the complexity of the model used to generate the training dataset.

To ensure the aforementioned requirements in terms of model parameterization and computational efficiency, we propose to combine patient-specific FE simulations with deep learning techniques. Within this thesis work, our main contribution is the use of deep neural networks for real-time numerical simulations of nonlinear deformations.

This manuscript is divided in two main parts. The first part is dedicated to generic real-time simulations with deep neural networks. The first two chapters establish the theoretical foundations of the problems we are looking at. We first present the mechanical formulation of our problem and the finite-element method used for its numerical resolution. In a next chapter, the fundamentals of artificial neural networks are introduced and we give a brief explanation on how such networks are trained. In the third chapter, we present the main contribution of our work: the U-Mesh framework. It consists of a data-driven deep neural network that learns the desired biomechanical model to predict complex nonlinear deformations in real-time on simple geometries.

The second part of this thesis deals with adapting the finite-element method and the deep learning algorithms to patient-specific modeling. We look at the importance of correct model parameterization and we describe a stochastic assimilation method to identify the patient-specific parameters. Chapter 4 describes how to perform patient-specific modeling using Kalman filtering. Chapter 6 concerns medical robotics. A neural network for robotic force estimation is trained with patient-specific finite-element simulations. The objective is to estimate forces during robotized intravitreal injections using a neural network for image classification. Chapters 7 and 8 extend the U-Mesh framework to patient-specific geometries. We demonstrate its potentiality in *computer-assisted interventions*, in particular augmented liver surgery and US-guided breast biopsy. In the last chapter of this manuscript, we propose several strategies in order to adapt the network's prediction to patient-specific properties.

Part I

Real-time numerical simulations

Chapter 2
Biomechanical Problem Formulation

2.1 Elasticity equations for soft tissue simulation

A biomechanical model is a mean to represent the physical behavior of an organ whose mechanical properties and boundary conditions are difficult to determine. In order to construct such a model, we explain the physical phenomena considered in this work using the theory of continuum mechanics (assumption that the studied object fills entirely the space it occupies). The goal of a simulation is to predict the behavior of a deformable body from the knowledge of the external forces f_{ext} applied to the body. There are two types of forces undergone by an object: the external forces (such as gravity and contacts) and the internal forces that are given by the *stress tensor*. The internal forces maintain the coherence of the body: a deformable body cannot be infinitely stretched or compressed. Applying forces on a deformable body leads to the creation of constraints (stress) and deformations (strain) inside of it. Such quantities will be defined in next sections. From Newton's second law of motion we have that:

$$\mathbf{f}_{ext} + \mathbf{f}_{int} = m\mathbf{a}, \tag{2.1}$$

where $\mathbf{f}_{int}$ are the internal forces, m the body mass and $\mathbf{a}$ the acceleration. In the static case we can rewrite this equation as:

$$\mathbf{f}_{ext} = -\mathbf{f}_{int}. \tag{2.2}$$

2.1.1 A measure of deformation: the strain tensors

We observe a continuum solid occupying a volume Ω whose boundary is Γ. We consider a Lagrangian description of the movement meaning that the deformed state of a particle $\mathbf{p} \in \Omega$ at time t is given by

$$\mathbf{p} = \phi(\bar{\mathbf{p}}, t), \qquad \bar{\mathbf{p}} = \phi(\bar{\mathbf{p}}, 0) \tag{2.3}$$

where $\bar{\mathbf{p}}$ is the initial position of particle $\mathbf{p}$ and ϕ is the transformation undergone by the solid with $\phi : \Omega \times \mathbb{R}_+ \to \Omega$.

A naive approach to quantify the deformation between two infinitesimally close points $\bar{\mathbf{p_1}}$ and $\bar{\mathbf{p_2}}$ is to measure their respective distances in the initial and deformed configurations. In the initial configuration, such distance reads as

$$\mathbf{d\bar{p}} = \bar{\mathbf{p_2}} - \bar{\mathbf{p_1}} \tag{2.4}$$

and in the deformed configuration:

$$\mathbf{dp} = \mathbf{p_2} - \mathbf{p_1} = \phi(\bar{\mathbf{p_1}} + \mathbf{d\bar{p}}, t) - \phi(\bar{\mathbf{p_1}}, t) \tag{2.5}$$

where $\mathbf{p_1}$ and $\mathbf{p_2}$ are two infinitesimally close points in the deformed configuration and $\mathbf{dp}$ the distance between them. For each particle in the rest configuration, there exists a displacement vector u, $u : \Omega \times \mathbb{R}_+ \to \Omega$, describing its deformed position such that

$$u(\bar{\mathbf{p}}, t) = \phi(\bar{\mathbf{p}}, t) - \bar{\mathbf{p}}. \tag{2.6}$$

The displacement field u characterizes the movement of the solid. Nevertheless note that for a rigid deformation, u is different from zero even if the object is not deformed. Hence, a better definition of the deformation must be introduced.

Let us introduce the deformation gradient in $\mathbb{R}^{3\times3}$:

$$\mathbb{F} = \nabla u + \mathbb{I}, \tag{2.7}$$

where $\mathbb{I} \in \mathbb{R}^{3\times3}$ is the identity matrix and ∇u is the gradient of u. $\mathbb{F}$ describes the variation of the distance between the particles in all the directions. Equation (2.5) can be rewritten as

$$\mathbf{dp} = \mathbb{F}\mathbf{d\bar{p}}. \tag{2.8}$$

The square of distance $\mathbf{dp}$ leads to

$$\begin{aligned}(d\mathbf{p})^2 = \mathbf{dp} \cdot \mathbf{dp} &= (\mathbb{F}\mathbf{d\bar{p}}) \cdot (\mathbb{F}\mathbf{d\bar{p}}) \\ &= (\mathbb{F}\mathbf{d\bar{p}})^T(\mathbb{F}\mathbf{d\bar{p}}) \\ &= \mathbf{d\bar{p}}^T\mathbb{F}^T\mathbb{F}\mathbf{d\bar{p}} \\ &= \mathbf{d\bar{p}}^T\boldsymbol{\epsilon}_c\mathbf{d\bar{p}},\end{aligned}$$

where $\boldsymbol{\epsilon}_c = \mathbb{F}^T\mathbb{F}$ is known as the right Cauchy-Green strain tensor. However, when the difference in angle between the rest and deformed configuration is null, $\boldsymbol{\epsilon}_c$ is not

null. Indeed, if $\mathbf{dp} = \mathbf{d\bar{p}}$, $\boldsymbol{\epsilon}_c = \mathbb{I}$. Therefore, we compute the difference between the squared distances

$$\begin{aligned}(\mathbf{dp})^2 - (\mathbf{d\bar{p}})^2 &= \mathbf{d\bar{p}}^T \boldsymbol{\epsilon}_c \mathbf{d\bar{p}} - (\mathbf{d\bar{p}} \cdot \mathbf{d\bar{p}}) \\ &= \mathbf{d\bar{p}}^T (\boldsymbol{\epsilon}_c - \mathbb{I}) \mathbf{d\bar{p}} \\ &= 2\mathbf{d\bar{p}}^T \boldsymbol{\epsilon}_g \mathbf{d\bar{p}},\end{aligned}$$

where $\boldsymbol{\epsilon}_g = \frac{1}{2}(\boldsymbol{\epsilon}_c - \mathbb{I})$ is known as the Green-Lagrange strain tensor satisfying $\boldsymbol{\epsilon}_g = 0$ when $\mathbf{dp} = \mathbf{d\bar{p}}$. It can be rewritten as:

$$\begin{aligned}\boldsymbol{\epsilon}_g &= \frac{1}{2}(\mathbb{F}^T \mathbb{F} - \mathbb{I}) \\ &= \frac{1}{2}\left((\nabla u + \mathbb{I})^T (\nabla u + \mathbb{I}) - \mathbb{I}\right) \\ &= \frac{1}{2}\left(\nabla u^T \nabla u\right) + \frac{1}{2}\left(\nabla u^T + \nabla u\right).\end{aligned}$$

The Green-Lagrange tensor consists of a linear component $\boldsymbol{\epsilon}_{linear} = \frac{1}{2}\left(\nabla u^T + \nabla u\right)$ and a nonlinear one $\boldsymbol{\epsilon}_{nonlinear} = \frac{1}{2}\left(\nabla u^T \nabla u\right)$. By linearization of $\boldsymbol{\epsilon}_g$, for small deformations $\boldsymbol{\epsilon}_{nonlinear}$ can be neglected and $\boldsymbol{\epsilon}_g$ can be approximated by:

$$\boldsymbol{\epsilon} := \boldsymbol{\epsilon}_{linear} = \frac{1}{2}\left(\nabla u^T + \nabla u\right). \tag{2.9}$$

The infinitesimal strain tensor $\boldsymbol{\epsilon}$ considerably simplifies the equations. However, as soon as the deformations get large, this linearization is no longer valid and the complete $\boldsymbol{\epsilon}_g$ must be considered.

2.1.2 A measure of internal forces: the stress tensor

We now need a way to account for the internal forces of an object that constrain its deformation (avoiding infinite stretch or compression for instance). Let $\mathbf{t}_n$ be the stress vector at point $\mathbf{p}$ whose normal to the surface is $\mathbf{n}$:

$$\mathbf{t}_n = \frac{\mathbf{f}}{dS}, \tag{2.10}$$

where $\mathbf{f}$ is the average force applied on the infinitesimal surface dS. The stress vector $\mathbf{t}_n$ depends on the normal $\mathbf{n}$, hence it does not completely describe the impact of the force in the whole volume. If we consider an infinitesimal volume dV, for example a tetrahedron (see Figure 2.1), the force at the wider face whose normal is $\mathbf{n}$ is expressed as:

$$\mathbf{f} = \mathbf{t}_n dS, \tag{2.11}$$

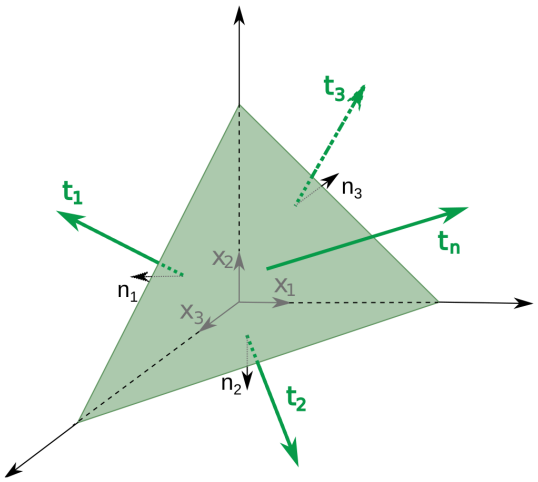


Figure 2.1: Infinitesimal volume tetrahedron. The stress vectors $\mathbf{t}_1$, $\mathbf{t}_2$, $\mathbf{t}_3$ and $\mathbf{t}_n$ are linked to the normals $-\mathbf{x}_1$, $-\mathbf{x}_2$, $-\mathbf{x}_3$ and $\mathbf{n}$.

where dS is the area of the considered face. On the other faces, the force is expressed as:

$$\mathbf{f}_i = -\mathbf{t}_i dS_i \quad i \in \{1, 2, 3\}, \tag{2.12}$$

where $\mathbf{t}_i$ is the stress vector and dS_i is the area of the i-th face of the tetrahedron. The minus sign comes from the normals that are equal to the negative of the coordinate axis. Let $\mathbf{f}_v dV$ be the volume force inside the tetrahedron. From the *conservation of linear momentum* we have that:

$$\mathbf{f} + \mathbf{f}_1 + \mathbf{f}_2 + \mathbf{f}_3 + \mathbf{f}_v dV = \rho dV \frac{d\mathbf{v}}{dt}, \tag{2.13}$$

where ρ is the mass density, $\mathbf{v}$ is the velocity of the tetrahedron. Using equations (2.11), (2.12) and expliciting dV, Equation(2.13) can be rewritten as:

$$\mathbf{t}_n dS - \mathbf{t}_1 n_1 dS - \mathbf{t}_2 n_2 dS - \mathbf{t}_3 n_3 dS + \mathbf{f}_v hc dS = \rho hc dS \frac{d\mathbf{v}}{dt}, \tag{2.14}$$

where h is the height of the tetrahedron, c is a constant independent of h and (n_1, n_2, n_3) are the coordinates of the normal $\mathbf{n}$. If we devide by dS and make h tend to 0 we get that:

$$\mathbf{t}_n - \mathbf{t}_1 n_1 - \mathbf{t}_2 n_2 - \mathbf{t}_3 n_3 = 0. \tag{2.15}$$

In other words, if the stress vectors acting on three faces of normal parallel to the coordinate axes is known, we can compute the stress vector for any normal direction $\mathbf{n}$:

$$\mathbf{t}_n = \begin{pmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \mathbf{t}_3 \end{pmatrix} \begin{pmatrix} n_1 \\ n_2 \\ n_3 \end{pmatrix} = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{pmatrix} \begin{pmatrix} n_1 \\ n_2 \\ n_3 \end{pmatrix} = \boldsymbol{\sigma}_c \cdot \mathbf{n} \tag{2.16}$$

where $\boldsymbol{\sigma}_c$ is called the Cauchy stress tensor.

There exist other definitions for the stress tensor such as the first Piola-Kirchhoff tensor $\boldsymbol{\sigma}_p$:

$$\boldsymbol{\sigma}_p = \mathbf{J} \cdot \boldsymbol{\sigma}_c \cdot \mathbb{F}^{-T}, \tag{2.17}$$

where $\mathbf{J} = \det(\mathbb{F})$ is the matrix allowing to pass from the rest to the deformed configuration. However, $\boldsymbol{\sigma}_p$ is not symmetric and is based on the deformed configuration. On the contrary, the second Piola-Kirchhoff tensor defines the stress based on the rest configuration:

$$\boldsymbol{\sigma}_s = \mathbf{J}\mathbb{F}^{-1} \cdot \boldsymbol{\sigma}_c \cdot \mathbb{F}^{-T}. \tag{2.18}$$

2.1.3 Constitutive law: linking stress and strain

There is a relation between the strain tensor $\boldsymbol{\epsilon}$ and the stress tensor $\boldsymbol{\sigma}$ that depends on the mechanical properties of the material. Such function translates the link between the forces applied on the considered body and its deformation. Even if it can be very complex, for small displacements we can assume its linearization using Hooke's constitutive law.

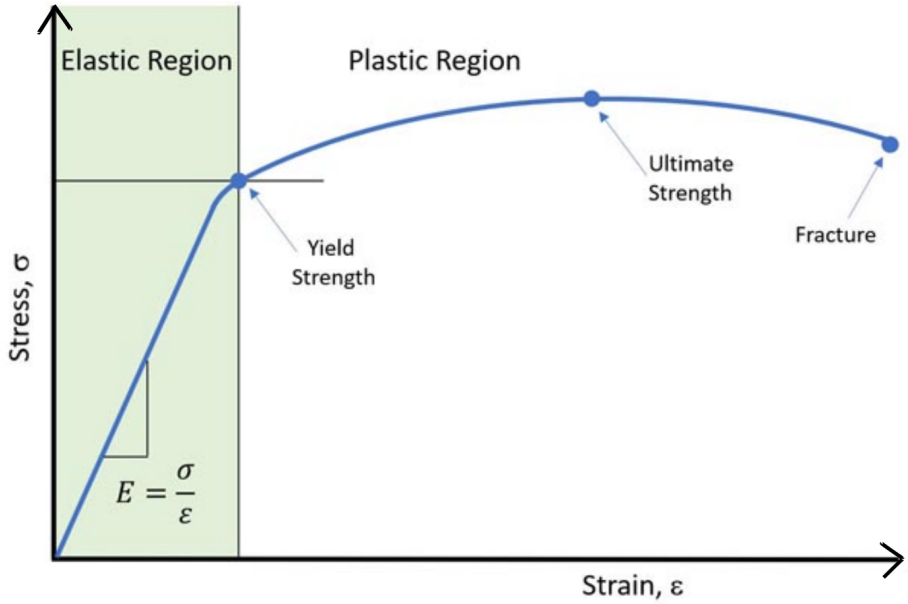


Figure 2.2: Linearization of the relation between stress and strain. Source: `www.simsolid.com`

Linear elasticity: Hooke's law The simplest relation is given by *Hooke's law* claiming that the elongation is directly proportional to the force. This linear constitutive equation relates two symmetric tensors:

$$\boldsymbol{\sigma} = \mathbf{C}\boldsymbol{\epsilon}, \tag{2.19}$$

where $\mathbf{C}$ is the *constitutive matrix* (see [Slawinski, 2007] for detailed equations). If the material studied is isotropic, such equation can be written as:

$$\boldsymbol{\sigma} = 2\mu\boldsymbol{\epsilon} + \lambda tr(\boldsymbol{\epsilon})\mathbb{I}. \tag{2.20}$$

where λ and μ are the so-called *Lamé coefficients*, which in the present case are related to the material properties of the soft body (introduced later in Equation(2.41)). Note that Hooke's law is linear in $\boldsymbol{\epsilon}$ but also in ∇u because we used the linearized strain tensor. If we do not work with small displacements, we need to take account for nonlinearities of the displacement and of the material, and the relation between strain and stress is no longer linear in ∇u (cf Figure 2.2). The material should therefore be modeled using an hyperelastic constitutive law such as the Saint Venant-Kirchhoff, Ogden, Neo-Hookean or Mooney-Rivlin models.

Hyperelasticity: the Saint Venant-Kirchhoff model If we rewrite the Equation (2.20) with $\boldsymbol{\epsilon}_g$ instead of the linearized $\boldsymbol{\epsilon}$, we obtain the Saint Venant-Kirchhoff model where $\boldsymbol{\sigma}$ is not linear in ∇u anymore. It is the simplest and most efficient extension of a linear elastic material to the nonlinear regime. In the following chapters, we will most of the time use the Saint Venant-Kirchhoff model.

It goes without saying that the constitutive law can be made more complex in order to model a higher degree of nonlinearity but the computational needs also increase. For a given mesh resolution and configuration (constant material properties and boundary conditions) the computation time needed for solving linear elasticity equations can be much smaller than that needed for hyperelasticity equations. Therefore, it is crucial to adapt the choice of the constitutive model to the considered deformations (to avoid using unnecessarily complex and computationally expensive models).

2.1.4 Forces and weak formulation

In the previous sections, we have defined the strain tensor linking the deformation and the displacement. Then, we have described the stress tensor translating the internal forces undergone by the soft body. Finally we have characterized the relation function between the strain and the stress tensors. To balance the internal forces with the external forces, we can express the force equilibrium over the computational domain Ω such that:

$$div(\boldsymbol{\sigma}) + f_{ext} = \rho\ddot{u}, \tag{2.21}$$

where ρ is the density of the material and f_{ext} the external forces. The Equation (2.21) is known as the strong formulation. It represents the fundamental principle of dynamics at each particle of the body. As we will only consider static problems

throughout this manuscript, we can set the second derivative of the displacement to zero. Hence, Equation (2.21) reads as:

$$-\, div(\boldsymbol{\sigma}) = f_{ext}. \tag{2.22}$$

To ensure the uniqueness of the solution to this equation, we must impose Dirichlet boundary conditions to function u on a subset Γ_D of the boundary of the domain Ω such that

$$u(x,t) = u_D,\ \forall x \in \Gamma_D, \tag{2.23}$$

where u_D is a constant. Also surface forces f^s are applied on a subset Γ_N of the boundary Γ (the so-called Neumann conditions) such that

$$\boldsymbol{\sigma}(x).\vec{n}(x) = f^s,\ \forall x \in \Gamma_N. \tag{2.24}$$

The boundary conditions will be further detailed in Section 2.3.2.

In order to distinguish the forces that are applied at the boundaries of the body such as pressure and contacts (denoted f^s) from the volume forces such as gravity (denoted f^v), we introduce the weak formulation of Equation (2.22). The weak form consists of multiplying the strong form by a test function v (verifying $v = 0$ on Γ_D) and integrating over the domain Ω. It is given by Equation (2.28) using Green's formula:

$$-\int_\Omega div(\boldsymbol{\sigma}).v\ d\Omega = \int_\Omega f_{ext}.v\ d\Omega \tag{2.25}$$

$$\Leftrightarrow \int_\Omega \boldsymbol{\sigma} : \boldsymbol{\epsilon}(v)\ d\Omega - \int_\Gamma \boldsymbol{\sigma}.\vec{n}.v\ d\Gamma = \int_\Omega f_{ext}.v\ d\Omega \tag{2.26}$$

$$\Leftrightarrow \int_\Omega \boldsymbol{\sigma} : \boldsymbol{\epsilon}(v)\ d\Omega - \int_{\Gamma_N} f^s.v\ d\Gamma - \int_{\Gamma_D} \boldsymbol{\sigma}.\vec{n}.v\ d\Gamma = \int_\Omega f^v.v\ d\Omega \tag{2.27}$$

$$\Leftrightarrow \underbrace{\int_\Omega \boldsymbol{\sigma} : \boldsymbol{\epsilon}(v)\ d\Omega}_{\text{Material stiffness}} - \underbrace{\int_{\Gamma_N} f^s.v\ d\Gamma}_{\text{Boundary conditions}} = \underbrace{\int_\Omega f^v.v\ d\Omega}_{\text{Volume forces}} \tag{2.28}$$

We remark that the spaces of the solution u and the test function v are different: $u = u_D$ on Γ_D, whereas $v = 0$ on Γ_D. In the following, for the sake of simplicity, we will set $u_D = 0$.

These equations must be applied to all the points that form the body but this would take an infinite amount of time. Hence, we need a method that approximates the solution to this formulation: the *finite-element method* that will be explained in the next section.

2.2 The Finite-Element Method

The equations of the previous section must be integrated all over the domain Ω, which leads to an infinite number of equations. The *finite-element method* (FEM) is a numerical method that gives a numerical approximation of a partial derivatives equation discretizing the object using nodes connected by elements. Positions inside the elements are interpolated from the element nodes. Continuum mechanics produces a stiffness matrix for each element that relates nodal displacements to nodal forces. FEM has interesting properties for our application domain such as accuracy and robustness. However, its computation time and sensitivity to the mesh resolution raise several challenges in the context of real-time simulation of organs. In this section, we will present the FEM for the simulation of deformable objects.

2.2.1 Domain discretization

The first step is the domain discretization. It consists of generating a mesh composed of simple geometrical shapes (for example triangles or quadrilaterals in 2-D or tetrahedra or hexahedra in 3-D) that composes a non-gap and no intersection subdivision of the computational domain. We then obtain a finite number of points that are linked by elements. The main idea is to solve the equations at these elements and then approximate the solution over the entire domain Ω by a linear combination of element-wise continuous polynomial functions called the *shape functions*. The order of the polynomial gives the order of the element (usually linear elements or quadratic elements).

Tetrahedral finite-element meshes are usually preferred over hexahedral meshes as the latter are extremely difficult to generate automatically, and its generation takes several orders of magnitude longer than tetrahedral mesh generators [Shepherd et al., 2008]. However, in this manuscript we will mainly use hexahedral finite element meshes. The main motivation for this choice is that we want to use the FE simulations to train *Convolutional Neural Networks* that work on regular grids (see Chapter 4 for details). Moreover, hexahedral meshes offer several benefits over tetrahedral finite element meshes. They present smaller error and less elements are usually needed to reach a given accuracy, and tetrahedral elements can exhibit a too stiff behavior [Shepherd et al., 2008]. Moreover, trilinear hexahedral elements ($\mathbb{Q}_1$ elements) provide similar accuracy and convergence characteristics as quadratic tetrahedral elements ($\mathbb{P}_2$ elements) [Cifuentes et al., 1992, Benzley et al., 1995]. For all these reasons, we will use trilinear hexahedral elements in our simulations.

To compute accurately the deformation of the considered object, the finite-element mesh needs to be sufficiently fine (to avoid discretization errors). The more we refine, the better the solution but at the cost of higher computational resources. Therefore,

a compromise must be found in order to obtain the desired accuracy within an acceptable amount of time.

2.2.2 Shape functions

In this section, we will explain how to build the finite-element space in which we will search the numerical solution. For the sake of readability, we will only present the case of a linear constitutive law associated to the small displacement regime, since handling the linearized version is similar. We will now express the *shape functions* (also known as basis functions) that allow us to interpolate the value of the displacement field through each element based on the nodal values. Throughout this manuscript we only consider linear or trilinear shape functions e.g. first degree polynomials. Note that we could use higher degree polynomials but at the cost of higher computation times. For triangular or tetrahedral elements, the $\mathbb{P}_1$ nodal *shape function* N_i at node i reads as:

$$N_i(x, y, z) = \sum_{j+k+l \leq 1} \alpha_{jkl} x^j y^k z^l, \tag{2.29}$$

where α_{jkl} are real coefficients defined such that $N_i = 1$ at node i and $N_i = 0$ at other nodes. For quadrilateral or hexahedral elements, the $\mathbb{Q}_1$ *shape function* reads as:

$$N_i(x, y, z) = \sum_{j,k,l \leq 1} \alpha_{jkl} x^j y^k z^l, \tag{2.30}$$

where, again, α_{jkl} are real coefficients defined such that $N_i = 1$ at node i and $N_i = 0$ at other nodes.

At any point p, the discretized displacement u_h can be expressed as:

$$u_h(p) = \sum_{i=1}^{N} N_i(p).u_{hi}, \tag{2.31}$$

where u_{hi} is the displacement at node i and N is the number of nodes in the mesh (excluding those on the Dirichlet boundary).There is one shape function N_i for each node i in the mesh and such function must have a local support and be piece-wise continuous.

For readability reasons, we rewrite $\sum_{i=1}^{N} N_i$ as $\sum_k N_k$ in the following equations.

2.2.3 Build the system matrices

In order to write the system in a matrix form, we first need to write Equation (2.28) in its discrete form. The problem we aim at solving reads as:

$$\exists ? u_h \in V_{h,0} \text{ such that } \int_\Omega \boldsymbol{\sigma}(u_h) : \boldsymbol{\epsilon}(v_h)\ d\Omega - \int_{\Gamma_N} f_h^s.v_h\ d\Gamma = \int_\Omega f_h^v.v_h\ d\Omega,\ \forall v_h \in V_{h,0}. \tag{2.32}$$

u_h is the discretized version of the displacement u and is given by:

$$u_h(p) = \sum_k N_k(p).u_{hk}. \tag{2.33}$$

f_h^s and f_h^v are the discretized forms of the surface and volume forces expressed with the shape functions. They read as follows:

$$f_h^s(p) = \sum_k N_k(p).f_{hk}^s, \tag{2.34}$$

$$f_h^v(p) = \sum_k N_k(p).f_{hk}^v. \tag{2.35}$$

Space $V_{h,0}$ is the discretization of the Hilbert space $H_0^1(\Omega)$.

If in addition, we apply the Galerkin theorem suggesting to replace the test function by the shape function $N_n(p)$, by linearity we get that:

$$\sum_k u_{hk} \left(\int_\Omega \boldsymbol{\sigma}(N_k(p)) : \boldsymbol{\epsilon}(N_n(p))\ d\Omega \right) - \sum_k f_{hk}^s \left(\int_{\Gamma_N} N_k(p).N_n(p)\ d\Gamma \right) = \sum_k f_{hk}^v \left(\int_\Omega N_k(p).N_n(p)\ d\Omega \right).$$

Let us define the stiffness matrix $\mathbf{K}$:

$$\mathbf{K} = \begin{pmatrix} \int_\Omega \boldsymbol{\sigma}(N_1(p)) : \boldsymbol{\epsilon}(N_1(p))\ d\Omega & \dots & \int_\Omega \boldsymbol{\sigma}(N_1(p)) : \boldsymbol{\epsilon}(N_N(p))\ d\Omega \\ \vdots & \vdots & \vdots \\ \int_\Omega \boldsymbol{\sigma}(N_N(p)) : \boldsymbol{\epsilon}(N_1(p))\ d\Omega & \dots & \int_\Omega \boldsymbol{\sigma}(N_N(p)) : \boldsymbol{\epsilon}(N_N(p))\ d\Omega \end{pmatrix}, \tag{2.36}$$

and the displacement vector U_h:

$$U_h = \begin{pmatrix} u_{h1} \\ \vdots \\ u_{hN} \end{pmatrix}. \tag{2.37}$$

Let us also define the volume and surface force vectors as:

$$F_h^s = \begin{pmatrix} f_{h1}^s \\ \vdots \\ f_{hN}^s \end{pmatrix} \text{ and } F_h^v = \begin{pmatrix} f_{h1}^v \\ \vdots \\ f_{hN}^v \end{pmatrix}. \tag{2.38}$$

Then, the full system reads as:

$$\mathbf{K}U_h = F_h^s + F_h^v. \tag{2.39}$$

To derive the stiffness matrix we made the assumption of small displacements. However, most of the times, the stiffness matrix cannot be expressed explicitly due to the material nonlinearity (relation linking the stress and the strain tensors) or to the geometrical nonlinearity (relation between the strain tensor and the displacement). Hence, it is some times impossible to separate the material stiffness from the displacement u. In such case, the weak formulation will be solved iteratively thanks to the Newton-Raphson algorithm (presented in next section) and $\mathbf{K}$ will be replaced by the associated tangent stiffness matrix.

Note that in the case of small displacements, we suppose that the matrix $\mathbf{K}$ does not change over time (meaning that we can compute its inverse once for all at the beginning of the simulation thus accelerating the computations considerably). For more complex materials, $\mathbf{K}$ will need to be recomputed at every simulation step. Intuitively, the matrix $\mathbf{K}$ explains the influence of a point on the other points of the mesh. It will be used to know the internal forces generated over the nodes by their displacement.

2.2.4 Solving the system of equations

Equation (2.39) represents the static system that aims at studying the equilibrium states of the deformation. The solution is the displacement for which the internal forces are equal to the external forces applied to the object. In this case, any linear system solver could be used for solving the equation. In the nonlinear case, we can obtain the solution of the system by iterating in the Newton-Raphson method for example. Basically we build a sequence in which we make the displacement vary until the difference of the forces vanishes. From an initial displacement u^0, we try to find a correction δ_u^n after n iterations that balances the linearized set of equations:

$$\dot{\mathbf{K}}^{n-1}(u^{n-1})\delta_u^n = \mathbf{r}(u^0 + \delta_u^{n-1}) + f \tag{2.40}$$

where $\dot{\mathbf{K}}$ is the tangent stiffness matrix and $\mathbf{r}$ is the internal elastic force vector. At each iteration, both the matrix $\dot{\mathbf{K}}$ and the vector $\mathbf{r}$ need to be computed, and the linear system needs to be solved. Since the convergence of the Newton-Raphson method is only valid for a displacement u^0 near the solution, large external loads must be applied by small increments and can require a large number of iterations to converge.

If we were to consider dynamics, in Equation (2.21), ü would be different from 0. Dynamic systems aim at studying the deformations that depend on the time,

thus taking into account the acceleration of the system. We want to know all the transitory states of the movement. In this case, a temporal integration scheme has to be used before the linear system solver. For that, we must define a time-step h and a time integration scheme in order to define the new positions in terms of the old ones. There are two possible strategies: explicit temporal integration or implicit integration. We will not go any further in the time integration since we only deal with static problems throughout this manuscript.

2.3 Model parameterization

2.3.1 Material properties

The proper modeling of an object requires the identification of the constitutive law (presented in Section 2.1.3) and the related material properties. These parameters play a major role in the estimation of the displacement field. Therefore, their understanding and identification is crucial for the accuracy of the method.

However, identifying such parameters can be very difficult in practice, mostly for soft tissues. Indeed, the typical approach for estimating them relies in traction experiments where different known forces are applied on a sample of the material. The sample elongation and the change in its cross section are measured in order to estimate the Young's modulus E and the Poisson's ratio ν. The Young's modulus defines the ratio of stress to strain. It is a measure of the stiffness of the material. The Poisson's ratio characterizes the compressibility of the material (change in the volume). While the Young's modulus can take any positive value, the Poisson's ratio is positive and smaller than 0.5. Incompressible materials (such as soft tissues) have a Poisson's ratio close to 0.5. The Lamé coefficients λ and μ are theoretical material-dependent quantities that are in practice obtained form the values of E and ν through the following relations:

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \qquad \mu = \frac{E}{2(1+\nu)}. \tag{2.41}$$

Note that such definition is only valid for linear elasticity and for some cases of hyperelasticity (such as the Saint-Venant Kirchhoff model that is the only hyperelastic law used in this manuscript).

For biological tissues, the traction experiments are controversial because the biological samples are examined *ex vivo*. Therefore, the mechanical properties obtained may differ from the *in vivo* properties. Other methods such as elastography [Sarvazyan et al., 1998, Muthupillai et al., 1995, Xu et al., 2007] or Bayesian filtering [Peterlík et al., 2017] attempt to determine these parameters *in vivo*. Whereas

in the first part of this manuscript we will consider average material properties, in the second part we will identify them with Bayesian filtering (see Chapter 5).

2.3.2 Boundary conditions

In order to guarantee the uniqueness of the solution of Equation (2.22), boundary conditions must be imposed on the unknown function u. The boundary conditions describe the effect of the exterior environment on the considered object. There are two types of boundary conditions: Dirichlet and Neumann. We recall that Γ is the boundary of the domain Ω. Let Γ_D be the subpart of Γ on which Dirichlet conditions are applied and Γ_N the subpart on which Neumann conditions are applied (such that $\Gamma_D \cap \Gamma_N = \emptyset$). Dirichlet conditions specify the value of the function u on the domain boundary, while the Neumann conditions impose a traction at the boundary. In the first part of this manuscript, we assume the Dirichlet conditions on Γ_D are known *a priori*, while Neumann boundary conditions on Γ_N can change at any time step. In the context of soft tissue simulation, the correct estimation of boundary conditions is a non-trivial problem [Nikolaev et al., 2018]. For instance, the structures attaching the organs (such as ligaments or muscles) are difficult to locate and characterize based on medical images. Therefore, in the second part of this manuscript, we will propose a way to characterize them using Bayesian filtering (similarly to material properties).

2.4 Conclusion

In this chapter we have presented elastic biomechanical models for soft-tissue simulation in order to estimate the displacement field u that is the quantity of interest. Moreover, the finite-element method is depicted with an emphasis on the discretization choices considering the further use of convolutional neural networks for displacement field estimation. In addition, we introduced the model parameters and boundary conditions that play a major role in the accuracy of the prediction as we will see in Chapter 5.

Chapter 3

Artificial Neural Networks: Fundamentals

3.1 Components of an artificial neural network

3.1.1 Definition

An artificial neural network (ANN) is a sequence of equations that approximates the underlying relationships in a given data set. Such a system learns the patterns in the data by analyzing examples, generally without receiving any task-specific rules. An ANN is an application f, nonlinear with respect to its parameters θ that associates an output y to an input x such that $y = f(x, \theta)$. A neural network (NN) can be used for solving regression or classification problems and it can be trained with labeled (supervised learning) or unlabeled data (unsupervised learning). The universal approximation theorem [Cybenko, 1989, Hornik, 1991] states that a feed forward neural network made of artificial neurons can approximate any real-valued continuous function on compact subsets of $\mathbb{R}^n$.

In Figure 3.1 a simple NN is depicted. The input layer appears in orange and consists of five inputs. It receives the information supposed to explain the function to be approximated. Then, there are two hidden layers with two and five artificial neurons respectively. A hidden layer is an intermediate layer allowing to model the nonlinear process. In such a feed forward NN, the outputs of each layer are the inputs to the next layers. The output layer appears in green and is the NN prediction.

3.1.2 Artificial neuron and activation functions

Each artificial neuron consists of a function h_j of the input $x = (x_1, ..., x_d)$, weighted by a vector of connection weights $w_j = (w_{j,1}, ..., w_{j,d})$, completed by a neuron bias b_j, and associated to an activation function ϕ such that

$$y_j = h_j(x) = \phi(< w_j, x > + b_j). \tag{3.1}$$

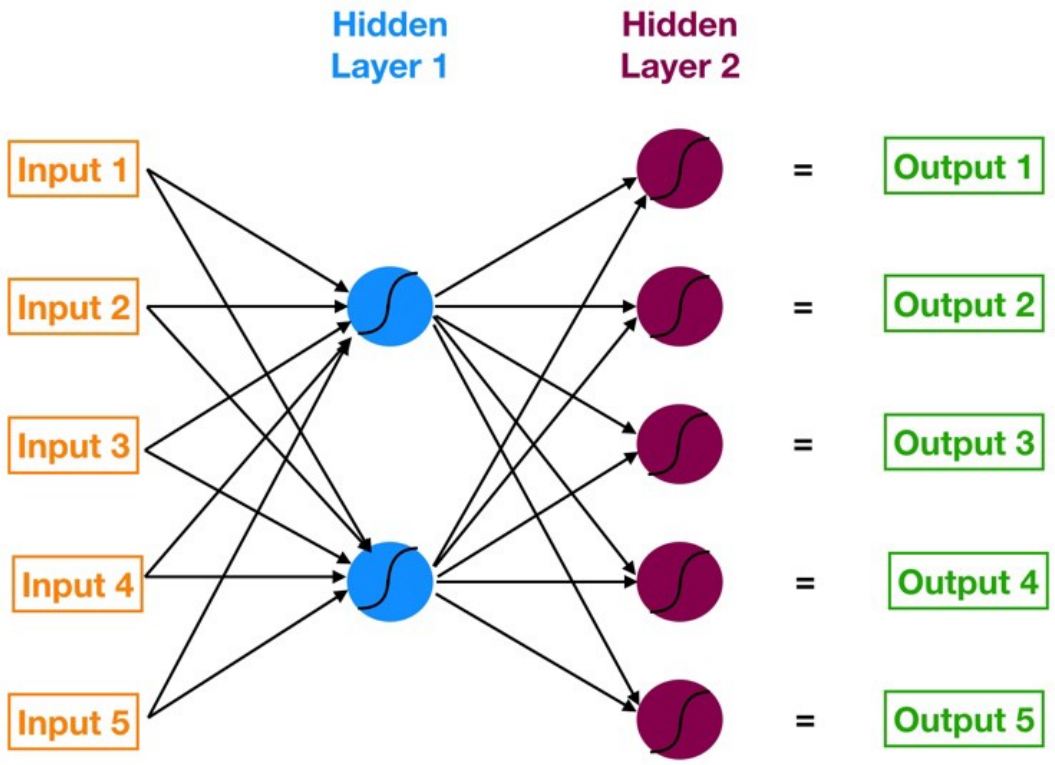


Figure 3.1: Multi-layer perceptron with five inputs, five outputs and two hidden layers of neurons. Source: `https://towardsdatascience.com`

The activation function ϕ introduces the nonlinearity and converts the signal entering a neuron into an output signal. The bias b_j allows to shift the activation function curve up or down leading to greater learning opportunities for the network. An activation function can be linear or nonlinear. Here are some examples of activation functions:

- The identity function
$$\phi(x) = x,$$
- The sigmoid function
$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}},$$
- The hyperbolic tangent
$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}},$$
- The rectified linear unit (ReLU)
$$\text{ReLu}(x) = max(0, x),$$

- The hard threshold function
$$\phi_\beta(x) = \mathbb{1}_{x \geq \beta}.$$

Historically, the Sigmoid was the most used activation function as it is differentiable and translates the input ranged in $[-\infty; +\infty]$ to the range $[0; 1]$. However, the exponential function is computationally expensive and the Sigmoid has the problem of vanishing gradients (the same problem arises for the hyperbolic tangent). The problem of vanishing gradients is inherent to the backpropagation optimization. As we move backwards in the NN, the gradients tend to get smaller and smaller leading to difficulties in training the first layers. In particular, training the model with vanishing gradients can be very time consuming and lead to low accuracy in the prediction.

On the other hand, the ReLU activation function does not suffer from vanishing gradients. It is cheap to compute and it accelerates the convergence of the Stochastic Gradient Descent algorithm (that will be explained later). However, as the ReLU function and its derivative are equal to 0 for negative values, no information can be obtained in that case for the concerned neuron. To overcome this issue, a small positive bias can be added to the activation function to ensure that each neuron is active:
$$\phi(x) = \max(x, 0) + \epsilon \min(x, 0), \tag{3.2}$$
where ϵ is either a fixed parameter set to a small positive value, or a parameter to estimate (e.g. to learn).

ReLU is used only within hidden layers. For the output layer, a different activation function can be used depending on the type of problems we are dealing with (regression or classification). In binary classification, the output is a prediction of $\mathbb{P}(Y = 1/X)$ since it ranges in $[0, 1]$ so the Sigmoid activation function can be considered. However, for multi-class classification, the output layer has one neuron per class i, giving a prediction of $\mathbb{P}(Y = i/X)$. The sum of all these values has to be equal to 1. Hence one can use the Softmax activation function that reads as:
$$\text{softmax}(z)_i = \frac{e^{(z_i)}}{\sum_j e^{(z_j)}}. \tag{3.3}$$

On the other hand, for regression problems one can use as activation of the output layer the hyperbolic tangent, a linear function or the identity function (meaning no activation function).

3.2 Specific types of neural networks

In this section we will present the two types of neural networks used in this thesis: the multi-layer perceptron with fully connected layers and convolutional neural networks.

3.2.1 Multi-layer perceptron

A multilayer perceptron (MLP) is a feedforward neural network composed by at least three layers of neurons where the output of a neuron of a layer becomes the input of a neuron of the next layer. In a MLP, each neuron of a layer is linked to all the neurons of the next layer but has no link with the neurons of the same layer. The mathematical formulation of a MLP with L hidden layers reads as follows:
We set $a^{(0)}(x) = x$.
For $k = 1, ..., L$ (hidden layers)

$$\begin{aligned} z^{(k)}(x) &= b^{(k)} + W^{(k)} a^{(k-1)}(x), \\ a^{(k)}(x) &= \phi(z^{(k)}(x)). \end{aligned} \tag{3.4}$$

For $k = L + 1$ (output layer)

$$\begin{aligned} z^{(k)}(x) &= b^{(k)} + W^{(k)} a^{(k-1)}(x), \\ a^{(k)}(x) &= \psi(z^{(k)}(x)). \end{aligned} \tag{3.5}$$

where ϕ is the activation function of the hidden layers and ψ is the output layer activation function. Superscript $^{(i)}$ stands for the i-th layer. At each step, $W^{(k)}$ is a matrix with number of rows the number of neurons in the layer k and number of columns the number of neurons in the layer $k - 1$.

The number of parameters of such a network is equal to the sum of the multiplications of the number of neurons between each consecutive layer, which in practice can be very high. Moreover, since the input of an MLP must be a vector, depending on the dimension of the problem, this approach might be inefficient when dealing with images or higher dimension inputs.

3.2.2 Convolutional neural networks

In this section we will present Convolutional neural networks (CNN). For the sake of simplicity we will explain them in 2 dimensions. The main advantage of CNNs is that they work directly with tensors of any dimension. As depicted in figure 3.2, a CNN is made of several types of layers: convolutional layers, pooling layers and fully connected layers. In the following, each type of layer will be detailed.

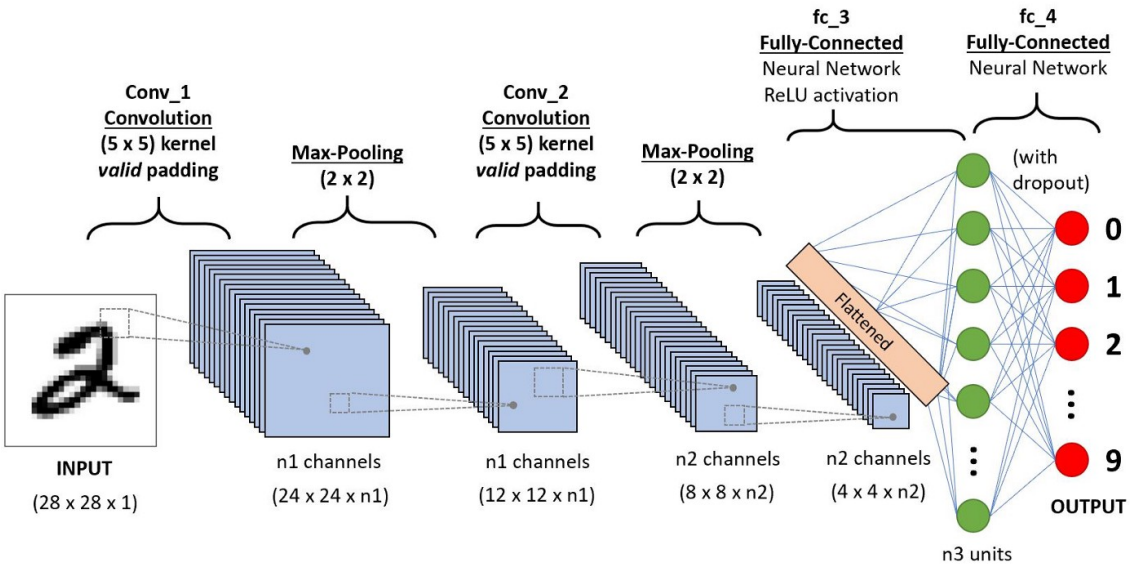


Figure 3.2: CNN to classify handwritten digits. Source: `https://towardsdatascience.com`

Convolutional layer A convolutional layer is composed of 2 main parts: the object and the filters. Filters or kernels are matrices of learnable weights that extract the features from an input signal. The discrete convolution of two functions f and g reads as:

$$(f * g)(x) = \sum_t f(t)g(x+t). \tag{3.6}$$

For a 2D image I, the 2D-convolution with convolution kernel K is given by:

$$(K * I)(i,j) = \sum_{m,n} K(m,n)I(i+n, j+m). \tag{3.7}$$

The convolution kernel is dragged on the image. At each position, we compute the convolution between the kernel and the patch of image over which the kernel is hovering. Then, the kernel slides by a number s of pixels (s being the stride) till it parses the complete image. To control the size of the output, zero padding can be added around the image (margin of zeros).

After the convolution operations, a ReLu activation function is generally added. If K is a convolution kernel of size $k \times k$ and x is a patch of the image of size $k \times k$ pixels, the activation is obtained by sliding the $k \times k$ window and computing $z(x) = \phi(K * x + b)$ where b is a bias.

Several convolution layers can follow one another where the output of a convolution becomes the input of the next one.

Pooling layer Pooling layers reduce the spatial dimension of the convolved features by taking the mean or the maximum on patches of the input (mean-pooling or max-pooling). These layers also act on small patches so there is also a pooling stride. This step allows to extract the dominant features which are rotational and positional invariant.

Fully connected layer A CNN generally ends with one or various fully connected layers (e.g. perceptron layers) in order to learn nonlinear combinations of the high-level features extracted by the convolutional layers. The output of the convolutions must be flattened into a column vector before being fed to the dense layers.

In the same manner as for MLP, depending on the problem, a final activation function can be added (Softmax for classification for example).

Important remark CNNs have been widely used to extract features from images consisting of 2D or 3D matrices of values that are spatially connected. Note that the use of convolutional filters expects the input to be encoded in a **grid structure** as such filters have a spatial representation of the inputs. This is a key point to keep in mind for next chapters, where we will compute the deformations of an object using CNNs. Indeed, we cannot directly feed a CNN with the mesh of an object. Instead, we will first have to express the physical quantities of interest in a regular grid structure. We will explain in detail this procedure in chapters 4 and 7.

3.3 Training a neural network

Once the architecture of the network is set, the optimal parameters (weights and biases) must be found. To do that we perform a gradient descent minimization of a loss function that is chosen by the user. The general procedure for training a network consists of 7 main steps:

1. Initialize the weights with close to zero values;
2. Feed the network with an input;
3. *Forward propagation*: neurons are activated depending on their weights. Spread activations until the prediction is computed;
4. Compute the error between the prediction and the expected value using the *loss function*;
5. *Backpropagation*: spread the error back into the network. Update the weights such that the error is minimized and adjust the *learning rate*;

6. Repeat steps 1 to 5 and adjust weights after a batch of samples;
7. When the entire data set has been seen by the network, it is called an *epoch*. Repeat more epochs.

In next section we will present the different loss functions available and some variations of the gradient descent algorithm. Then we will explain the backpropagation algorithm.

3.3.1 Loss functions

To learn, the network must know the committed error over each batch. To do that, the gap between the prediction $f(x, \theta)$ and the expected value y is measured using a loss function $\mathcal{L}$. There are several options for the loss function depending on the considered problem. A common way for finding the parameters of a network is to maximize the likelihood (or the log likelihood), which is equivalent to minimizing the loss function (which is the opposite of the log likelihood). For a distribution $\mathcal{D}$ of pairs (X, Y), the expected loss function reads as

$$\mathcal{L}(\theta) = -\mathbb{E}_{(X,Y)\sim\mathcal{D}}\left(\log(p_\theta(Y/X)\right). \tag{3.8}$$

If the model is Gaussian, that is to say if $p_\theta(Y/X = x) \sim \mathcal{N}(f(x, \theta), I)$, maximizing the likelihood is equivalent to minimizing the quadratic loss

$$\mathcal{L}(\theta) = \mathbb{E}_{(X,Y)\sim\mathcal{D}}\left(\|Y - f(X, \theta)\|^2\right). \tag{3.9}$$

In the case of binary classification, maximizing the log likelihood is equivalent to minimizing the cross-entropy. Setting $f(X, \theta)) = p_\theta(Y = 1/X)$,

$$\mathcal{L}(\theta) = -\mathbb{E}_{(X,Y)\sim\mathcal{D}}\left[Y\log(f(X, \theta)) + (1 - Y)\log(1 - f(X, \theta))\right]. \tag{3.10}$$

This loss function is well adapted to the Sigmoid activation as the logarithm prevents small values of the gradient. Finally, for a multi-class classification problem, a generalization of the previous loss function to k classes is considered

$$\mathcal{L}(\theta) = -\mathbb{E}_{(X,Y)\sim\mathcal{D}}\left[\sum_{j=1}^{k} \mathbb{1}_{Y=j}\log(Y = j/X)\right]. \tag{3.11}$$

Alternatively, one could encode the knowledge of the data domain in the cost function. For instance, in the case of modeling biomechanics with a NN, the cost function can correspond to the strong formulation of the elasticity equations [Raissi et al., 2019]. We will not enter into detail for this type of loss functions as they are not consider throughout this manuscript.

3.3.2 Minimization with Stochastic gradient descent

To minimize a cost function, the gradient descent algorithm is one of the most popular methods. In the case of neural networks, the minimization is performed over the network parameters (weights and biases). For the sake of visual clarity, in the following the biases $\{b_j\}_{j=1,...r}$ are contained in the weights $\{w_j\}_{j=1,...n}$.

The gradient descent canonical formula reads as:

$$\theta := \theta - \alpha.\nabla\mathcal{L}(\theta), \tag{3.12}$$

where $\theta = [w_1, w_2, ..., w_n]^T$ is the parameters vector, α is the learning rate and $\nabla\mathcal{L}(\theta)$ is the gradient of the cost function such that $\nabla\mathcal{L}(\theta) = \left(\frac{\partial\mathcal{L}}{\partial w_1}, \frac{\partial\mathcal{L}}{\partial w_2}, ..., \frac{\partial\mathcal{L}}{\partial w_n}\right)$.

The learning rate α is an hyperparameter of the minimization. It determines the step size at each iteration while moving towards a minimum. It impacts the learning speed and the convergence. A large value of α might lead to fluctuation around an optimum thus making the optimization diverge (can result in underfitting). On the contrary, a too small value of α can slow down the convergence speed and converge to a local minimum. In this case, the network can adjust too well to the training data set and it is no longer able to generalize (overfitting). The optimal value of α cannot be analytically calculated for a given model on a given dataset. It is usually advised to adapt it during training but there is however no general rule. It is usually acknowledged that it should be initialized at a large enough value (0.1 for instance) and progressively be reduced during successive iterations of the training process.

Gradient descent needs the cost function to be convex. When this is not the case, taking the steepest slope is no longer enough (due to local minimums and saddle points). The risk is therefore to drastically slow down the backpropagation and to fall on a local minimum. The Stochastic gradient descent (SGD) algorithm addresses this issues. Its canonical formula is given by:

$$\theta := \theta - \alpha.\nabla\mathcal{L}(\theta; x^{(i)}; y^{(i)}). \tag{3.13}$$

Let $\{(x_n, y_n)\}_{n=1}^N$ be the training data set made of N samples. Instead of computing the gradient over all the samples of the data set, SGD computes the gradient over one randomly selected sample or subset of samples called a batch. At each step, m training samples are randomly selected without replacement and a combination of the m corresponding gradients is computed (usually the average). m is called the batch size. This process is repeated until the network has seen the entire training data set: this is called an epoch. The maximal number of epochs is usually fixed

in advanced and the training stops when this value is reached or when an accuracy criterion is satisfied (early stoping).

Algorithm 1: Stochastic Gradient Descent algorithm

Initialize N: size of the training set;
Initialize α: learning rate;
Initialize m: batch size;
Initialize nb_e: number of epochs;
for *epoch = 1 to* nb_e **do**
  for *batch = 1 to* $\frac{N}{m}$ **do**
    Take a random batch of size m without replacement $(x_i, y_i)_{i=1,...m}$;
    Forward pass;
    Compute the gradients with the backpropagation algorithm:
    $\tilde{\nabla}_\theta = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \mathcal{L}(f(x_i, \theta), y_i)$;
    Update the parameters $\theta^{new} = \theta^{old} = \alpha \tilde{\nabla}_\theta$;
  end
end

SGD is faster than standard (deterministic) gradient descent and provides more fluctuations of weights, which augments the probability of not being stuck in a non optimal local minimum.

Due to the high sensitivity of the SGD to the learning rate, variations of the algorithm have been proposed with adaptive learning rates. One of the most famous is the Adam optimizer [Kingma and Ba, 2014] with adaptive moment estimation. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.

3.3.3 Backpropagation

The backpropagation of the gradient was proposed by [Le Cun, 1988] as an efficient way to compute the gradient of a neural network through a method called chain rule. After each forward pass through a network, backpropagation performs a backward pass to adjust the network's parameters computing the gradient one layer at a time, iterating backward from the last layer to avoid redundant calculations of intermediate terms in the chain rule.

In this section we will explain how to compute the gradient of Equation (3.9) using the backpropagation algorithm. The empirical quadratic loss of Equation (3.9) is proportional to

$$\sum_{i=1}^{N} R_i(\theta), \tag{3.14}$$

with

$$R_i(\theta) = \sum_{k=1}^{K} (Y_{i,k} - f_k(X_i, \theta))^2, \tag{3.15}$$

where K is the size of the network's output. In a regression model, the output activation function ψ is generally the identity function, to be more general, we assume that

$$\psi(z_1, ..., z_K) = (g_1(z_1), ..., g_K(z_K)), \tag{3.16}$$

where $g_1, ..., g_K$ are functions from $\mathbb{R}$ to $\mathbb{R}$. Let us compute the partial derivatives of R_i with respect to the weights of the output layer. Recalling that

$$z^{(L+1)}(x) = b^{(L+1)} + W^{(L+1)} a^{(L)}(x), \tag{3.17}$$

we get

$$\frac{\partial R_i}{\partial W_{k,m}^{(L+1)}} = -2(Y_{i,k} - f_k(X_i, \theta)) g_k'(z_k^{(L+1)}(X_i)) a_m^{(L)}(X_i). \tag{3.18}$$

Differentiating now with respect to the weights of the previous layer

$$\frac{\partial R_i}{\partial W_{m,l}^{(L)}} = -2 \sum_{k=1}^{K} (Y_{i,k} - f_k(X_i, \theta)) g_k'(z_k^{(L+1)}(X_i)) \frac{\partial z_k^{(L+1)}(X_i)}{\partial W_{m,l}^{(L)}}. \tag{3.19}$$

with

$$z_k^{(L+1)}(x) = \sum_j W_{k,j}^{(L+1)} a_j^{(L)}(x), \tag{3.20}$$

$$a_j^{(L)}(x) = \phi\left(b_j^{(L)} + < W_j^{(L)}, a^{(L-1)}(x) >\right). \tag{3.21}$$

This leads to

$$\frac{\partial z_k^{(L+1)}(x)}{\partial W_{m,l}^{(L)}} = W_{k,m}^{(L+1)} \phi'\left(b_m^{(L)} + < W_m^{(L)}, a^{(L-1)}(x) >\right) a_l^{(L-1)}(x). \tag{3.22}$$

Let us introduce the notations

$$\begin{array}{l} \delta_{k,i} = -2(Y_{i,k} - f_k(X_i, \theta)) g_k'(z_k^{(L+1)}(X_i)) \\ s_{m,i} = \phi'\left(z_m^{(L)}(X_i)\right) \sum_{k=1}^{K} W_{k,m}^{(L+1)} \delta_{k,i}. \end{array} \tag{3.23}$$

Then we have

$$\begin{array}{l} \frac{\partial R_i}{\partial W_{k,m}^{(L+1)}} = \delta_{k,i} a_m^{(L)}(X_i), \\ \frac{\partial R_i}{\partial W_{m,l}^{(L)}} = s_{m,i} a_l^{(L-1)}(X_i). \end{array} \tag{3.24}$$

known as the backpropagation equations.

The values of the gradient are used to update the parameters in the gradient descent algorithm. At step $r+1$, we have:

$$\begin{aligned} W_{k,m}^{(L+1,r+1)} &= W_{k,m}^{(L+1,r)} - \alpha_r \sum_{i=1}^{m} \frac{\partial R_i}{\partial W_{k,m}^{(L+1,r)}} \\ W_{m,l}^{(L,r+1)} &= W_{m,l}^{(L,r)} - \alpha_r \sum_{i=1}^{m} \frac{\partial R_i}{\partial W_{m,l}^{(L)}}, \end{aligned} \tag{3.25}$$

where α_r is the learning rate that satisfies $\alpha_r \to 0$, $\sum_r \alpha_r = \infty$, $\sum_r \alpha_r^2 < \infty$, for example $\alpha_r = 1/r$. We use the backpropagation equations to compute the gradient by a two pass algorithm. In the forward pass, we fix the value of the current weights $\theta^{(r)} = (W^{(1,r)}, b^{(1,r)}, ..., W^{(L+1,r)}, b^{(L+1,r)})$, and we compute the predicted values $f(X_i, \theta^{(r)})$ and all the intermediate values $(z^{(k)}(X_i), a((k)(X_i) = \phi(z^{(k)}(X_i)))_{1 \leq k \leq L+1}$ that are stored. Using these values, we compute during the backward pass the quantities $\delta_{k,i}$, and $s_{m,i}$ and the partial derivatives given in Equation (3.23). We have computed the partial derivatives of R_i only with respect to the weights of the output layer and the previous ones, but we can go on to compute the partial derivatives of R_i with respect to the weights of the previous hidden layers.

In the backpropagation algorithm, each hidden layer gives and receives information from the neurons it is connected with. Hence, the algorithm is adapted for parallel computations. The computations of the partial derivatives involve the function ϕ', where ϕ is the activation functions. ϕ' can generally be expressed in a simple way for classical activation functions. The backpropagation algorithm is also used for classification with the cross-entropy but we will not enter into details for this aspect.

3.4 Conclusion

The objective of this thesis is to train neural networks for estimating physical quantities such as displacement fields or forces applied to an object. For this reason, in this chapter we have presented the different network architectures that we will use, and defined their constitutive elements for a better understanding of the approach that will be presented in next chapters. As seen in Section 3.3.3, the backpropagation equations involve a lot of tedious calculations that are fortunately automatically handled by frameworks such as Tensorflow [1] and Pytorch [2] that are both used during this thesis. In order to train the networks, large amounts of high fidelity data are needed which is the main difficulty of the approach, in particular when considering medical applications where acquiring thousands of real samples can be delicate. In the following chapters, we will demonstrate how to use FE simulations as accurate data generators to overcome the lack of real patient data. As seen in Section 3.2.2, the generated data must be expressed in a "network interpretative manner", namely

[1] https://www.tensorflow.org/
[2] https://pytorch.org/

a regular grid structure (for the case of CNNs). This is one of the main challenges that will be treated in next chapters.

Chapter 4
The U-Mesh Framework

4.1 Introduction and related work

As said in the general introduction, there are many applications in engineering where nonlinear deformations need to be simulated in real-time. Some important examples can be found in the field of medicine but also in other areas of mechanical engineering, such as training of complex industrial processes [Amundarain et al., 2004] or virtual prototyping [Barbic and James, 2008] just to name a few.

While there are different numerical strategies for solving equations associated with elastic materials, we only consider the *finite-element method* (FEM), for its accuracy and ability to simulate a large range of materials on potentially complex domains. However, obtaining real-time simulations with this method when considering nonlinear materials, becomes a real challenge, in particular if this has to be done on consumers level hardware rather than a high-end parallel computer.

In order to speed up FEM simulations, several techniques have been proposed. We review here only some of the main ideas that were proposed. First, since solving the system of equations resulting from the FEM discretization is usually the bottleneck of the computation, many works have focused on linear solvers. Domain decomposition methods are based on the "divide & conquer" paradigm. Such methods consist of splitting the global problem domain into smaller independent sub-domains, making the approach suitable for parallel computing. This allows to build efficient preconditioners even though additional computation is required to synchronize the solution between neighboring sub-domains. Under the right conditions, in particular if the number of processors of the computer matches the number of sub-domains, a super-linear speedup can be obtained [Haferssas et al., 2017]. This is, however, impossible to achieve when considering problems (even if relatively small) which need to be solved in real-time on consumers level hardware. This is mainly due to the limited number of cores (only 10 cores on the latest Intel i9 processor) and communication costs which are significant compared to the expected computation times (about 50 ms per time step for an interactive simulation).

Another option for speeding up simulation times is to lower the computational complexity of the problem through a reduction of the model's degrees of freedom. Depending on the problem, and acceptable loss of accuracy, it is possible to obtain speedups of several orders of magnitude. Proper Orthogonal Decomposition (POD) is one of the main model order reduction methods. POD techniques compute off-line the solution to several complete models and extract the modes that describe best the solution to the complete problem. Based on *a priori* knowledge of the solution, it encodes the high dimensional problem in a smaller subspace defined by a truncated basis of singular vectors. The dimension of such basis determines the ratio between accuracy of the method and computation times (the smaller the basis, the larger the error). In the context of real-time simulation of nonlinear solids, several examples of POD have been proposed. [Niroomandi et al., 2008] proposed a POD method to simulate the palpation of the cornea. Haptic feedback rates were achieved, but with a relative error of about 20%. If more accurate solutions are needed, the number of modes used in the POD can be increased at the cost of higher computation times. Thus, to keep the method numerically efficient in the case of nonlinear materials, hyperreduction can be used in order to further reduce the computation times while reducing the error [Ryckelynck, 2005]. [Goury and Duriez, 2018] applied the hyperreduced POD to control and simulate soft robots with very good accuracy in 25 ms per time step. However the POD may be in some cases insufficient to capture correctly the high degrees of nonlinearity that can be found for example in biological soft tissues as it relies on a linear combination of few basis vectors [Bhattacharjee and Matous, 2016]. To precisely account for nonlinearities it might be necessary to recompute the entire stiffness matrix which is burdensome and not always possible [Niroomandi et al., 2017]. Another model order reduction algorithm is the proper generalized decomposition (PGD), which, contrarily to POD, builds a reduced-order approximation without relying on the knowledge of the solution of the complete problem. PGD assumes that the solution of a multiparametric problem can be expressed as a sum of separable functions that are constructed by successive enrichment by invoking the weak form of the considered problem. An approach based on PGD is proposed in [Niroomandi et al., 2013] for the simulation of hyperelastic soft tissue deformation at high frequency. However, when the solution is non-separable, PGD offers no particular advantage over classical FEM techniques.

A last class of worth mentioning solutions consists of using the Graphics Processing Unit (GPU) as a particular type of parallel machine. Although each core of the GPU is very limited in its computational performance, the very high number of cores available (several thousand) makes it possible to obtain significant speedups on computationally heavy problems. For instance, NiftySim [Johnsen et al., 2015] is a GPU-based nonlinear finite element toolkit for the simulation of soft tissue biomechanics where speedups of 300x are obtained. SOFA [Allard et al., 2007] is an Open-source

Framework focused on real-time simulation of complex interactions with deformable structures, which provides GPU-compatible FEM codes [Comas et al., 2008].

Recently, machine learning started to revolutionize several fields (vision, language processing, image recognition, genomics) due to the continuously increasing amount of data available and the development of new algorithms and powerful GPUs. Deep learning, a class of machine learning methods based on learning data representations, as opposed to task-specific algorithms, has demonstrated strong abilities at extracting high-level representations of complex processes. With sufficient ground truth data, machine learning algorithms can map the input of a function to its output without any mathematical formulation of the problem, thus actuating like a black box. The term *black box* relates to the non-clarity in the way the model uses training data to reach particular conclusions. These ambiguities make it difficult to know the reasons why the system behaves the way it does. In this chapter we will put a lot of effort in clarifying such ambiguities by exploring connections of the chosen architecture with model order reduction techniques. To fulfill the need for ground truth data, since FEM can provide as much noise-free data as required, it seems interesting to train learning algorithms with such virtually generated data [Lorente et al., 2017, Luo et al., 2018, Roewer-Despres et al., 2018, Tonutti et al., 2017, Fetene et al., 2018, Runge et al., 2017]. For example [Lorente et al., 2017] proposed a machine learning approach for modeling the mechanical behavior of the liver during breathing in real-time. They trained several regression models using external displacement and elasticity parameters as input, and the FEM-based nodal displacements as output. Although they reached good accuracy their method is restricted to small displacements. [Roewer-Despres et al., 2018] proposed a preliminary work using a deep-autoencoder to approximate the deformations of a nonlinear muscle actuated object. They showed that their method produces lower reconstruction errors than the equivalently sized PCA model. However the computational gain of their method is not clear and it was limited to a very simple model and coarse mesh. [Tonutti et al., 2017] treated a simple problem using two different networks, one to predict the magnitude of the displacement and the other one to predict its direction. However, it seems that these two networks require a specific training for each node of interest, which could be prohibitive for large meshes. Moreover their model is limited to small deformations on relatively simple shapes, with restricted input forces. For instance, the considered displacements never exceed 5 mm (for an organ of size 20 cm) and only 11 nodes of the mesh are excited. On the contrary our approach can handle complex and complete volume deformations of arbitrary shapes with one single network for any force application point. All the cited references propose to train neural networks with FEM generated data for various purposes. However, none of them is justifying the choice of the network architecture used to do this. In this chapter, we explore connections of

the chosen architecture with model order reduction techniques in order to support interpretability and explainability to go beyond the black-box usage of neural networks.

The objective of our work, whose preliminary results are presented in this chapter, is to go beyond the state of the art on machine learning applied to computational mechanics. In particular we show that we can predict, in real-time, the shape of a nonlinear elastic structure with a very good accuracy using a deep network inspired by model order reduction methods. Our solution relies on a U-Net architecture trained on FEM-generated data sets; both aspects are presented in Section 4.2 while results and their comparison against a model order reduction algorithm are presented in Sections 4.3.

4.2 Displacement field estimation with deep neural networks

As mentioned previously, an important area of applications for real-time simulation of nonlinear material is in the field of computer-aided surgery where accuracy is also very important, but often left aside for the sake of rapidity. Achieving a better trade-off between accuracy and computation time is therefore mandatory to tackle more ambitious problems. From this chapter on, we propose a method that does not need such a compromise. It allows for extremely fast and accurate simulations by using an artificial neural network that partially encodes the stress-strain relation in a low-dimensional space. Such a network can learn the desired biomechanical model, and predict deformations at haptic feedback rates with very good accuracy. This section is divided in two main segments. First the selected network architecture is detailed, followed by our strategy to encode the stress-strain relationship and boundary conditions into the network and the data set generation used to train the network.

4.2.1 Network architecture

Formally, our network h is a parameterized function that accepts a $3 \times n_x \times n_y \times n_z$ tensor of input constraints $\mathbf{C}$ and produces a tensor of volume displacements $\mathbf{U_v}$ of the same size as output. The domain Ω is sampled by a 3-dimensional grid of resolution $n_x \times n_y \times n_z$. Practically speaking, the nodes of this grid match the nodes of the FEM mesh, although this is not required and an interpolation could be used instead. The tensor $\mathbf{C}$ represents the constraints applied over the surface boundary Γ of the domain. In particular, $\mathbf{C}$ can be seen as a traction $\mathbf{t}$ applied to Γ_N or as an imposed surface displacement $\mathbf{U_s}$ on Γ_D. The tensor $\mathbf{U_v}$ contains the

volumetric displacement at the nodes of the mesh in response to the constraint $\mathbf{C}$. In particular, each vector $\mathbf{C}[:, i, j, k]$ represents the constraint vector (c_x, c_y, c_z) applied over the node (i, j, k) of the grid. Similarly, the vector $\mathbf{U_v}[:, i, j, k]$ represents the displacement (u_x, u_y, u_z) of each node (i, j, k) of the grid.

Our problem consists of finding the function h that produces the best estimation of the displacement field given prescribed constraints $\mathbf{C}$ (in this chapter we only deal with contact forces e.g. traction). This is performed by minimizing the expected error over the distribution $\mathcal{D}$ of pairs $(\mathbf{C}, \mathbf{U_v})$:

$$\min_{\theta} \mathbb{E}_{(\mathbf{C},\mathbf{U_v})\sim\mathcal{D}} \left[\|h(\mathbf{C}) - \mathbf{U_v}\|_2^2 \right], \tag{4.1}$$

where θ is the set of parameters of the network h. In practice, the expectation of Equation (4.1) is approximated by Monte-Carlo sampling with a training set $\{(\mathbf{C}_n, \mathbf{U_v}_n)\}_{n=1}^N$ of N samples:

$$\min_{\theta} \frac{1}{N} \sum_{n=1}^{N} \|h(\mathbf{C}_n) - \mathbf{U_v}_n\|_2^2. \tag{4.2}$$

We build our training set by randomly applying forces on the mesh and running FEM simulations to produce corresponding displacements.

Let us characterize the architecture chosen for our network h. We propose to use the U-Net [Ronneberger et al., 2015], a modified fully convolutional network initially built for precise medical image segmentation. Note that the use of convolutional filters expects the input to be encoded in a grid, due to their spatial representation of the inputs. As depicted in Figure 4.1, the network is similar to an auto-encoder, with an encoding path to transform the input space into a low-dimensional representation, and a decoding path to expand it back to the original size. Additional skip connections transfer detailed information along matching levels from the encoding path to the decoding path.

The encoding path consists of k sequences of two padded $3\times3\times3$ convolutions ($k = 4$ in [Ronneberger et al., 2015]) and a $2\times2\times2$ max pooling operation (see Figure 4.1). Intuitively, each 3D convolution filter learns to isolate the different characteristics of the displacement field (orientation, direction, amplitude). At each step, each feature map doubles the number of channels and halves the spatial dimensions. We assume that the number of channels is directly related to the amount of detectable variations in the displacement field. In the bottom part there are two extra $3 \times 3 \times 3$ convolutional layers leading to a $(c \times 2^k)$-dimensional array. This feature space is intuitively similar to the Galerkin projection of the equations of motion onto the reduced space in POD, where the order of the singular vector truncation is equivalent to the number of neurons in the latent space. A difference however remains with

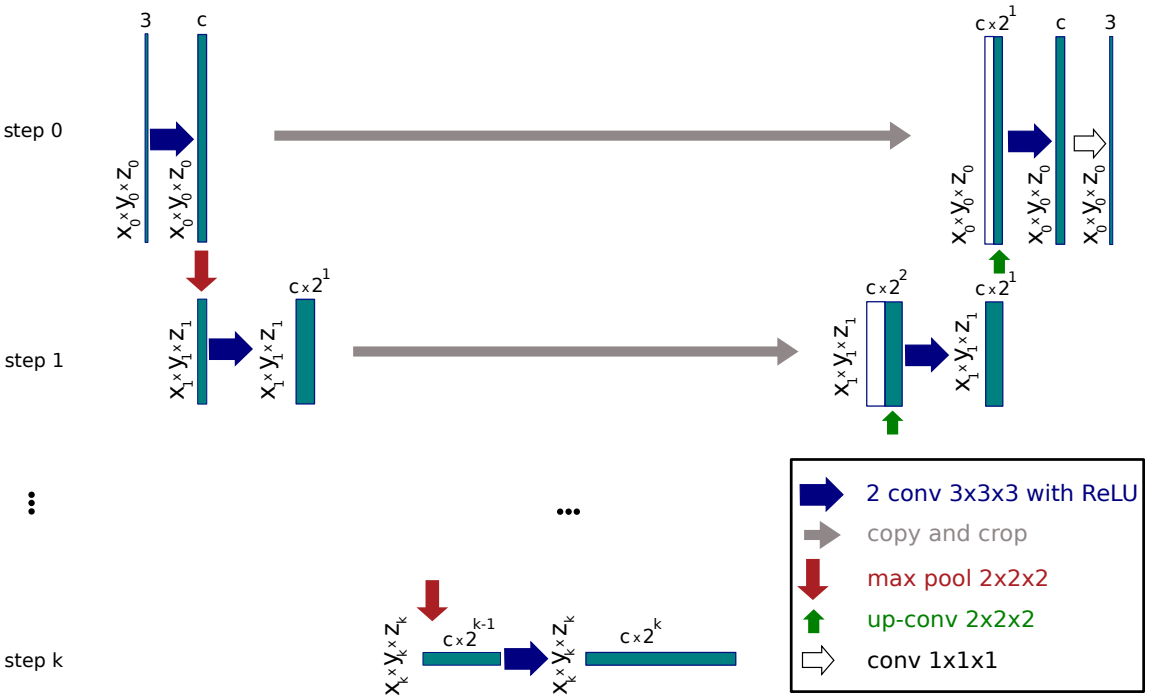


Figure 4.1: General network architecture for an object with a resolution of $x \times y \times z$ nodes, c channels in the first layer and k steps. Note that $x_0 \times y_0 \times z_0$ is the padded grid.

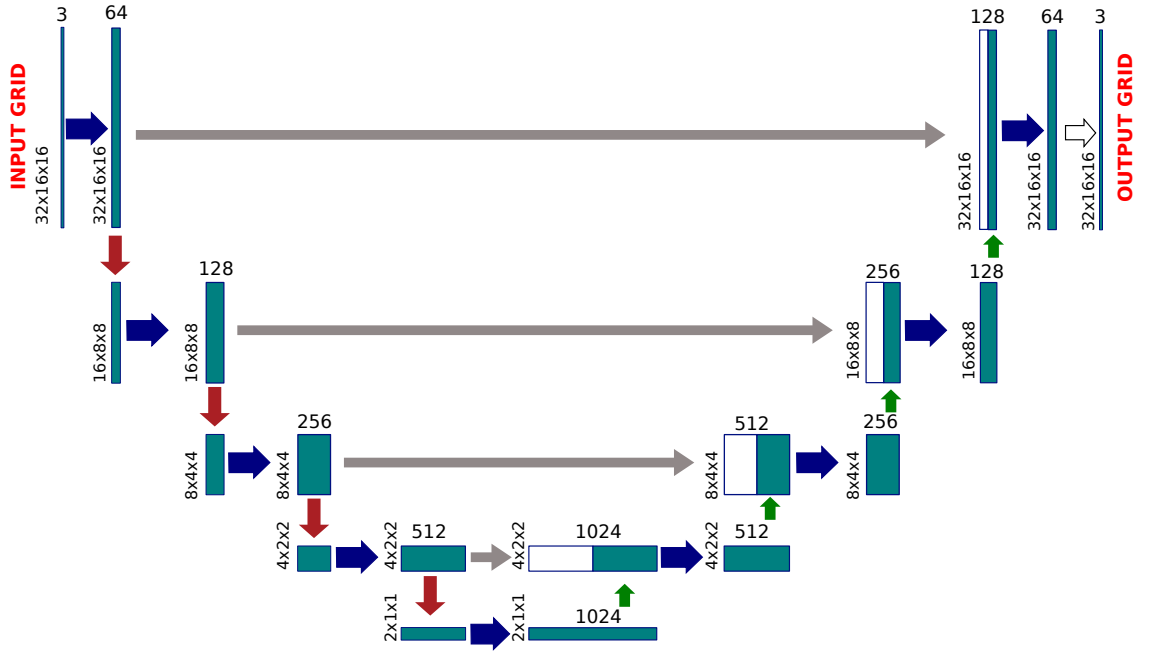


Figure 4.2: Network architecture for a beam with $28 \times 12 \times 12$ nodes, padded to $32 \times 16 \times 16$, 64 channels in the first layer and 4 steps (see Figure 4.1 for notations).

the presence of convolutional operations at each layer. In a symmetric manner, the decoding path consists of k sequences of an upsampling $2 \times 2 \times 2$ transposed convolutions followed by two padded $3 \times 3 \times 3$ convolutions. The features from the encoding path at the same stage are cropped and concatenated to the upsampled feature maps. At each step of the decoding path, each feature map halves the number of channels and doubles the spatial dimensions. There is a final $1 \times 1 \times 1$ convolutional layer to transform the last feature map to the desired number of channels of the output (3 channels in our case).

The number of steps k and the number of channels c control the accuracy of the prediction just like the number of singular vectors in POD. Higher values of k and c lead to a more complex network suitable for difficult meshes at the expenses of longer computational times for both training and prediction, and higher requirements of training data. We tested several values for k and c in order to select the most appropriate values for our experiments depending on the desired accuracy and the eventual time restrictions.

4.2.2 Offline training data generation

Without lack of generality, we consider the boundary value problem of computing the deformation of a hyperelastic material under both Dirichlet and Neumann boundary conditions described in Chapter 2. Hexahedral (H8) elements are used to discretize the computational domain. This choice is not only motivated by the good convergence and stability of such elements: hexahedral elements are also required for our convolutional neural network (see Section 4.2.1).

In order to train such a network, we build a data set of pairs $(\mathbf{C}, \mathbf{U_v})$ obtained with the FEM. Once we are given a 3D mesh with its corresponding constitutive law, material properties and boundary conditions, we perform multiple simulations by applying random forces to nodes of the object. After each simulation, the pair of applied forces and obtained deformation is stored as an element of the data set. To speed up the generation of the training and testing data sets, the linearized system of Equation (2.40) is solved using an iterative preconditioned conjugate gradient method [Shewchuk, 1994].

The variability of the data relies on the force magnitude, its direction and its application point. In this chapter, a force is applied on a local surface area Γ_N whose location varies such that the boundary of the computation domain is completely covered. The force direction is uniformly sampled on the unit sphere and the force magnitude is a uniform random value between 0 and 1. At each sample of the data set, one force is applied on a small region Γ_N. There are Λ samples for each Γ_N,

meaning that Λ different forces are applied per region. We can consider one force per sample or several forces applied simultaneously at different locations.

The network is trained minimizing Equation (4.2) with training data generated as explained above. The minimization is performed using the Adam optimizer [Kingma and Ba, 2014] (see Chapter 3 for details).

4.3 Results on synthetic objects

In this section, we perform a model selection over the space of hyperparameters k and c in order to find the combination that leads to the best results in a cantilever beam. Then we apply our method, with the selected hyperparameters, to two benchmark examples: a cantilever beam and an L-shaped object under point loads. All our experiments are performed in a GeForce 1080 Ti using a batch size of 4 and $100,000$ iterations for training. We use a PyTorch implementation of the U-Net. We recall that the batch size is the number of samples that are given to the network at each iteration of the minimization process.

4.3.1 Validation metrics

To assess the efficiency of our method, we perform a statistical analysis of the *mean norm error* e over a testing data set $\{(\mathbf{C}_m, \mathbf{U}_{\mathbf{v}m})\}_{m=1}^{M}$, which is built similarly as the training data set. Note that the training data set and the testing data set are disjoint. Let $\mathbf{U}_{\mathbf{v}m}$ be the ground truth displacement tensor for sample m generated using the FEM described in Section 2.2 of Chapter 2 and $h(\mathbf{C}_m)$ the U-Mesh prediction. The mean norm error between $\mathbf{U}_{\mathbf{v}m}$ and $h(\mathbf{C}_m)$ for sample m reads as:

$$e(\mathbf{U}_{\mathbf{v}m}, h(\mathbf{C}_m)) = \frac{1}{n}\sum_{i=1}^{n} |\mathbf{U}_{\mathbf{v}m}^{\ i} - h(\mathbf{C}_m)^i|. \tag{4.3}$$

where n is the number of degrees of freedom of the mesh. We compute the average $\overline{e}$ and standard deviation $\sigma(e)$ of such norm over the testing data set:

$$\overline{e} = \frac{1}{M}\sum_{m=1}^{M} e(h(\mathbf{C}_m), \mathbf{U}_{\mathbf{v}m}), \tag{4.4}$$

and

$$\sigma(e) = \sqrt{\frac{1}{M-1}\sum_{m=1}^{M} [e(h(\mathbf{C}_m), \mathbf{U}_{\mathbf{v}m}) - \overline{e}\,]^2}. \tag{4.5}$$

c	k	FSS	$\overline{e}$ in m	$\sigma(e)$ in m	pred_t in ms	train_t in min
64	2	256	0.0028	0.0008	2	24
16	4	256	0.0012	0.0009	3.2	40
32	4	512	0.0009	0.0008	3.2	40
64	3	512	0.0007	0.0007	2.5	80
64	4	1024	0.0007	0.0007	3.3	95
128	**3**	**1024**	**0.0007**	**0.0006**	**2.5**	**35**
64	5	2048	0.0006	0.0005	4	426
128	4	2048	0.0006	0.0005	3.45	320

Table 4.1: Error measures for a beam having 135 H8 elements. Rows are sorted in decreasing $\overline{e}$. The selected architecture appears in bold.

4.3.2 U-Mesh applied to a cantilever beam

We consider a deformable beam of size $4 \times 1 \times 1\,m^3$ subjected to fixed boundary on one end. The beam follows the Saint-Venant-Kirchhoff behavior described in Section 2.1.3 of Chapter 2 with a Young's modulus E of 500 Pa and a Poisson's ratio of 0.4, and is discretized with 135 H8 elements. To generate the data set, 100 forces are applied on each of the 64 nodes of the upper face (that is $\Lambda = 100$), up to a total of $6,400$ samples. 80% of the samples are used for training ($N = 5,120$) and the remaining 20% are kept for testing ($M = 1,280$). Using this data set, we select the best architecture by training several U-Nets with different combinations of hyperparemeters k and c. In Table 4.1 are reported the training and prediction times as well as $\overline{e}$ and $\sigma(e)$. As seen in this table, the higher the feature space size (FSS), the lower the errors. Prediction time is proportional to the depth of the network. Choosing the best model is a tradeoff between network performance and speed, and we will show that the selected hyperparameters lead to good results also on different problems. Choosing $k = 3$ and $c = 128$ seems to be a good compromise for our needs. The selected parameters appear in bold text in Table 4.1. For the selected set of parameters, e over the data set is equal to $0.0007 \pm 0.0006\,m$ for a maximal deformation of $0.724\,m$. In Figure 4.3, we show the sample with maximal error. We perform a sensitivity analysis of the method to the amplitude of deformation. The results are plotted in Figure 4.4. We perform a least squares line fit to find the relation between the maximal deformation and the mean norm error e. We can observe a very small sensitivity of the error e with respect to the deformation amplitude, and a very small error in the estimation of the displacement field in general. This shows an important characteristic of our method, in addition to its very limited computational cost.

In Table 4.2 are shown the prediction errors for 3 simultaneous input forces and their corresponding training and prediction times. In this scenario, the Young's modulus is set to 400 Pa and three forces are applied simultaneously on three different regions

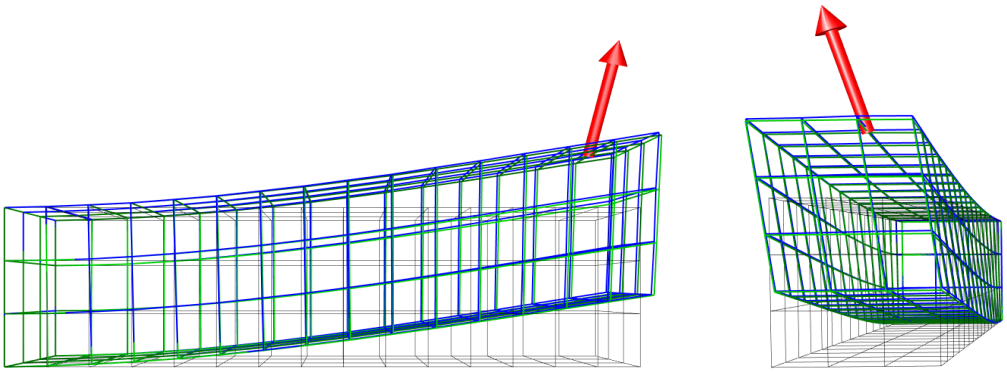


Figure 4.3: Front and side views of the maximal error sample obtained with a network having 3 steps and 128 channels. U-Mesh output is in blue and the reference is in green. The red arrow represents the force applied. Note that the difference between the two meshes is very small. The relative $l2$ norm at the tip of the beam is of 5.1% and the deformation amplitude is $0.45\,m$

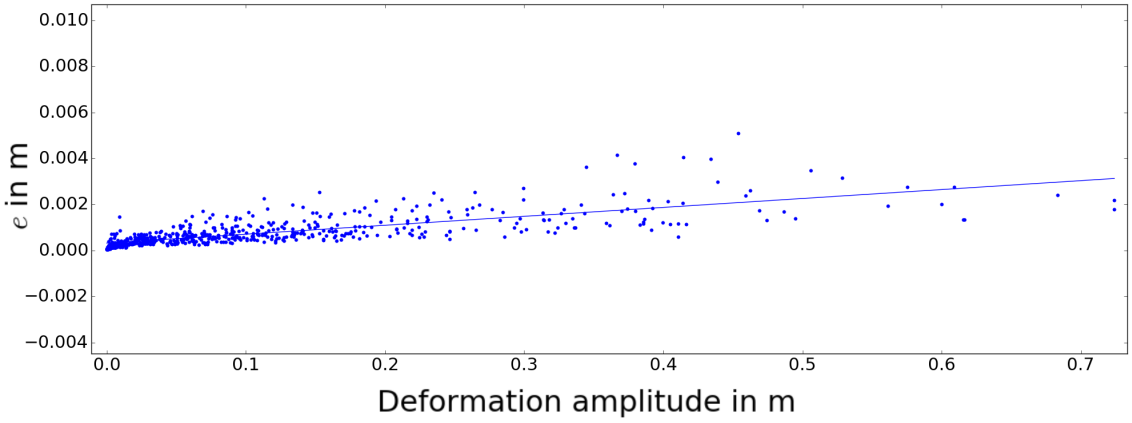


Figure 4.4: The point cloud represents the e for some randomly selected samples of the testing data set. The regression line of equation $y = 0.00352 \times x$ shows the low sensitivity of the U-Mesh to the deformation amplitude.

c	k	FSS	$\overline{e}$ in m	$\sigma(e)$ in m	pred_t in ms	train_t in min
64	4	1024	0.0007	0.0003	3.5	100

Table 4.2: Error measures for a beam having 135 H8 elements and three simultaneous force application.

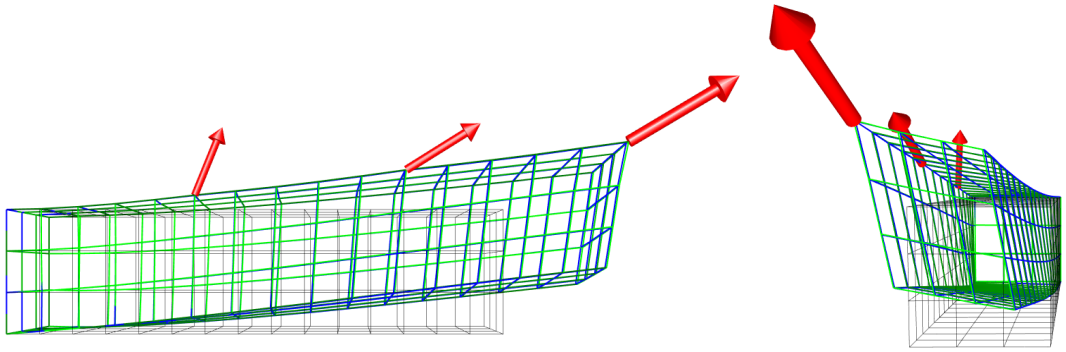


Figure 4.5: Largest deformation for 3 simultaneous force applications (side and front view for the same sample). U-Mesh output is in blue, the reference is in green and the rest shape is in grey.

of the upper face of the beam. In order to avoid mechanical coupling, such regions must be far enough from each other. There are $12,360$ possible combinations of nodes fulfilling the above stated condition. Only one force ($\Lambda = 1$) is applied per viable combination up to a total of $12,360$ samples ($N = 9,888$ samples for training and $M = 2,472$ samples for testing). We can see that the errors and the time needed for the prediction are comparable to that needed for only one force application. It is important to note that there are 2x more samples in this data set (than in previous ones) due to the large number of possible combinations. This explains the particularly low error in this case. The sample with maximal deformation ($1.0035\ m$) is shown in Figure 4.5. At the tip of the beam, the relative $l2$ norm is equal to 1.5%.

So far we have seen that U-Mesh is able to predict the displacement field due to one or several forces with high accuracy and in an extremely short amount of time. Nevertheless, FEM codes are also able to compute the solutions on such meshes in limited computation times, even for hyperelastic models. Hence, in order to put ahead the real contribution of our work, we test our method on a computationally expensive problem.

We consider the same beam as previously but this time discretized in $3,267$ H8 elements (see Figure 4.7). There are 336 nodes on the upper face and Λ is set to 100. Overall the data set has $33,600$ samples ($N = 26,880$ samples for training and $M = 6,720$ samples for testing). The U-Net is trained with the previously

c	k	FSS	$\overline{e}$ in m	$\sigma(e)$ in m	pred_t in ms	train_t in min
128	3	1024	0.0019	0.0018	3	210

Table 4.3: Error measures for a beam having 3267 H8 elements and one simultaneous force application.

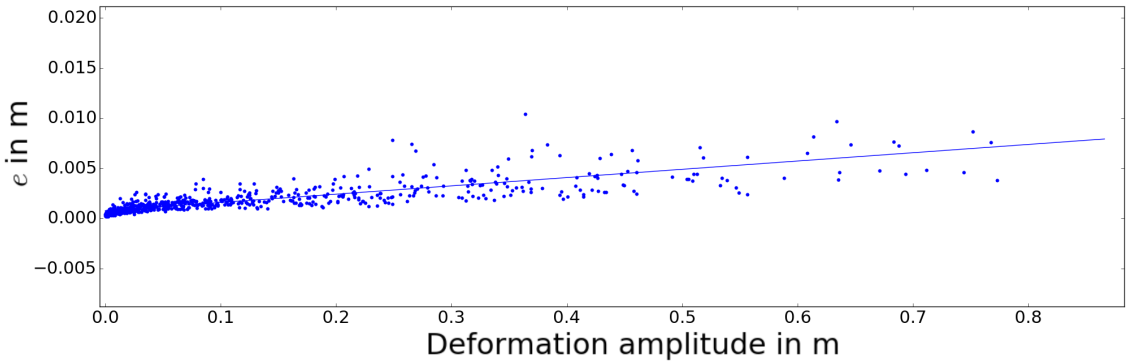


Figure 4.6: Sensitivity of e to the deformation amplitude for a $3,267$ H8-elements beam. The point cloud represents the values of e for randomly selected samples of the testing data set. The regression line of equation $y = 0.0083 \times x$ shows the low sensitivity of the U-Mesh to the deformation range. The average computation time is $3\,ms$.

selected parameters, that is, 128 channels for the first layer and 3 steps. The metrics computed over the testing data set are shown in Table 4.3. The most interesting result is the low prediction time (only $3\,ms$). A very optimized version of the Saint-Venant-Kirchhoff FEM using a Pardiso solver [Kourounis et al., 2018] that is among the most efficient solvers available, takes more than $300\,ms$ to solve such simulation. The speedup obtained with U-Mesh is of 100x. All the samples of the testing data set have an error bellow $0.0265\,m$ and an average error of $0.0019\,m$ for a maximal deformation of $1.011\,m$.

In the following paragraphs we will show that U-Mesh generalizes well on other geometries. In particular, we will see the performance of U-Mesh applied to an L-shaped object.

4.3.3 U-Mesh applied to an L-shaped object

We apply U-Mesh on an L-shape of size $28.424 \times 10 \times 40\ m^3$ discretized in 335 H8 elements. Since the U-Net requires a regular grid as input, the L-shaped object is embedded in a regular grid (with zero-padding). The Young's modulus is equal to 500 Pa and the Poisson's ratio is 0.4. To build the data set, external forces ranging from 0 to $40\,N$ are applied on the bottom face of the L (with $\Lambda = 100$) up to a

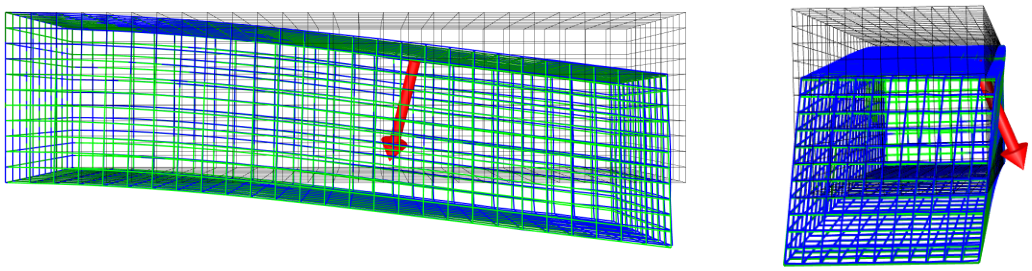


Figure 4.7: Beam mesh with 3267 H8 elements deformed with U-Mesh, front and side views. U-Mesh output is in blue, the reference is in green and the rest shape is in grey. The computation time is equal to $2.9\,ms$ for this sample. The relative $l2$ norm at the tip of the beam is 1.6% and the deformation amplitude is $0.4\,m$

c	k	FSS	$\overline{e}$ in m	$\sigma(e)$ in m	pred_t in ms	train_t in min
128	3	1024	0.00648	0.00493	3.2	97

Table 4.4: Error measures computed over the testing data set ($M = 1,200$ samples) for the L-shaped object.

total of $6,000$ samples ($N = 4,800$ and $M = 1,200$). We train a U-Net with 128 channels in the first layer and 3 steps. In Figure 4.8 are shown some samples of deformed L-shapes. The U-Mesh output is in blue and the reference solution is in green. In order to perceive the amount of deformation, the rest position is also shown (thin grey lines). The average and maximal errors are given in Table 4.4, and the prediction times are in the same range as for the beam scenario. The average error is equal to $0.00648\,m$ where the maximal deformation is $8.9016\,m$. The slope of the regression in Figure 4.9 shows that the increase of the error with the deformation amplitude is controlled. It is worth noting that the outliers of this graph (such as the one marked in red) still correspond to small errors (see Figure 4.10).

4.3.4 Comparison of U-Mesh and POD

In this section we compare the predictions made by U-Mesh to the simulations computed on a reduced model using POD. We used the POD code available at `https://github.com/SofaDefrost/ModelOrderReduction` that works as a plugin of the SOFA framework [Allard et al., 2007]. The POD consists of three phases. First, an offline phase where all the potential movements of the beam are sampled and stored in the so-called snapshot. This offline phase is the equivalent to the data generation phase in U-Mesh and is also computationally intensive since it performs many fine simulations. In a second phase, the snapshot space is condensed in a reduced

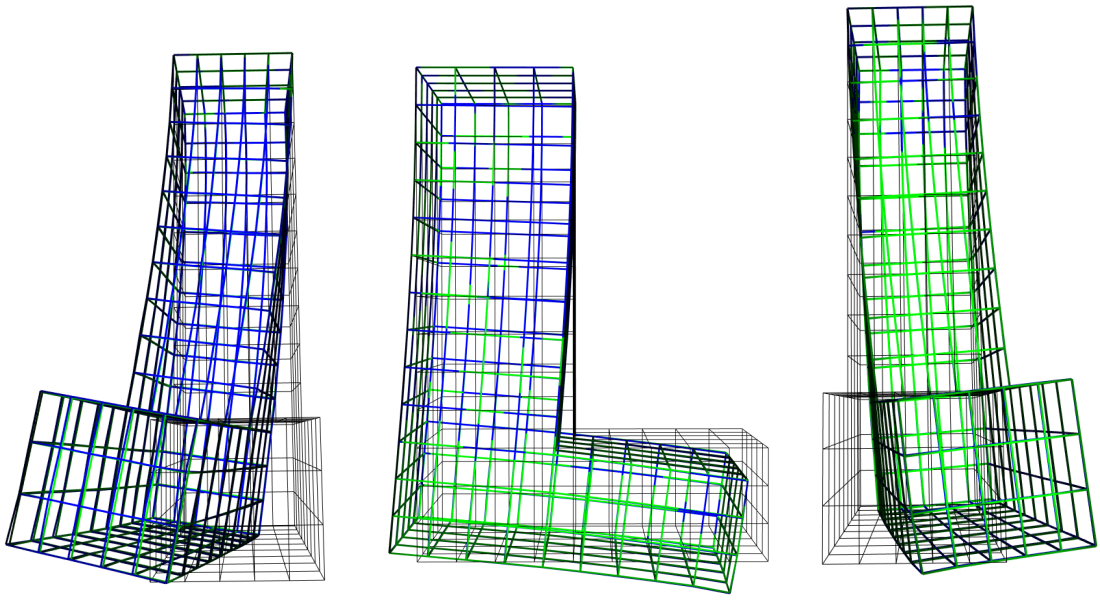


Figure 4.8: Samples of deformed L-shapes. In green is shown the reference solution, in blue the output of U-Mesh and in grey the rest shape.

basis using singular value decomposition and a truncation [Goury and Duriez, 2018]. This phase "corresponds" to the training of the U-Net and is generally faster. Finally, the resulting reduced model allows for faster simulations since there are fewer degrees of freedom. We applied the simple POD and the hyperreduced-POD (HPOD) to the fine beam depicted previously (see Figure 4.7) in order to compare the performance of POD and U-Mesh.

We first compare the computation times for a given accuracy. The truncature error of the POD was set such that the mean norm error obtained with POD is similar to the one obtained with U-Mesh. To reach the desired accuracy in the considered deformation range, 3 modes were preserved. Computation times are reported in Table 4.5. With the selected number of modes, POD is about 6 times faster than the full FEM model whereas U-Mesh is more than 200 times faster than the full FEM model.

Let us now compare the relative errors of the two methods for a given computation time. The fastest version of the reduced model is the one considering only one deformation mode and using hyperreduction. As presented in Table 4.6, HPOD can compute deformations in $5\,ms$ but with an error that is 14 times larger than the one produced by U-Mesh.

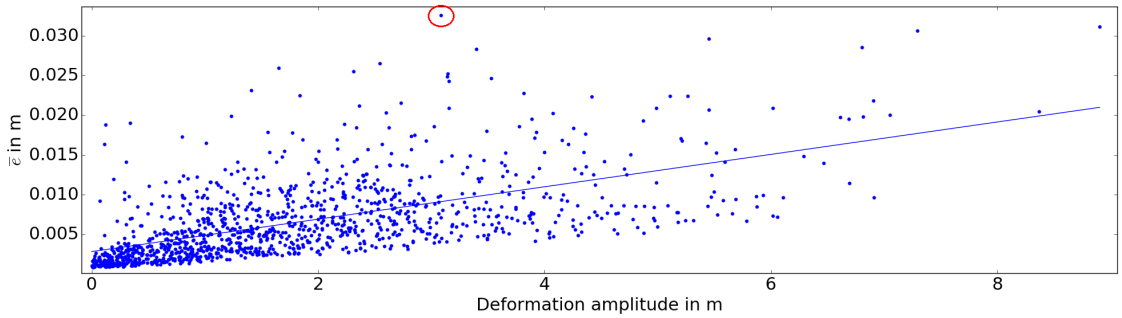


Figure 4.9: Sensitivity of e to the deformation amplitude for the L-shape. The point cloud represents the e all the samples of the testing data set. The regression line of equation $y = 0.002046 \times x$ shows the low sensitivity of the U-Mesh to the deformation range. Maximal error sample highlighted in red and shown in Figure 4.10.

	Prediction time in s	e in m
FEM	0.740	0.000
POD	0.120	0.006
U-Mesh	0.003	0.006

Table 4.5: Comparison at same error: computation times of the full FEM model, of the POD and of the U-Mesh, for the same mean norm error e. The number of modes used in the POD is 3.

4.4 Conclusion

In this chapter we have proposed a U-Net architecture that can learn the relation function between an input force and an output deformation for regular geometries and make predictions with high accuracy in a record amount of time. The take-home-message of this chapter is that for a given network architecture, the prediction time is nearly constant (and very short), regardless of the size of the problem. Furthermore the accuracy of the prediction, which depends on the quality and the size of the data set, is controllable since we generate this data. We believe that such an approach

	Prediction time in s	e in m
FEM	0.740	0.000
HPOD	0.005	0.084
U-Mesh	0.003	0.006

Table 4.6: Comparison at similar computation time: the mean norm errors of the full FEM model, of the HPOD and of the U-Mesh are given, for a computation time in the range of the millisecond. Only one mode is kept in the HPOD.

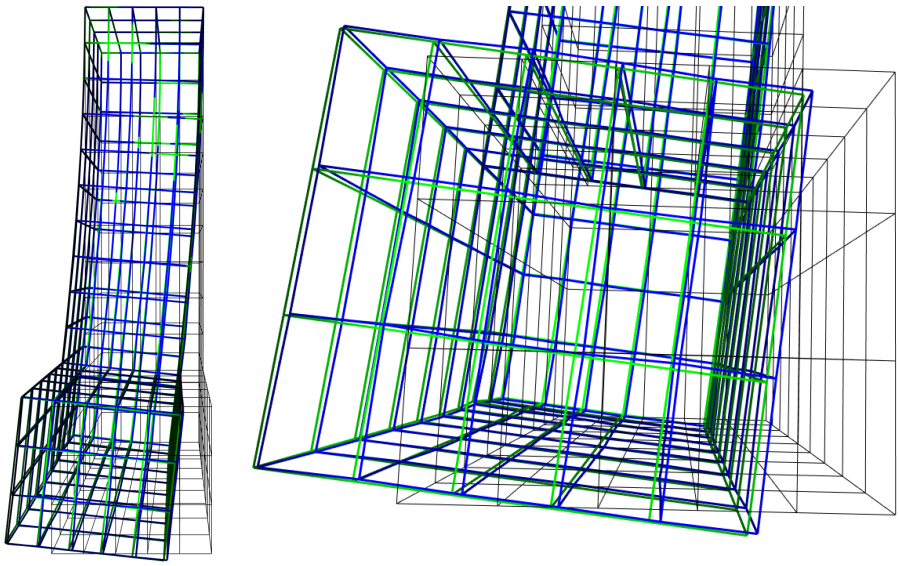


Figure 4.10: Maximal error sample ($e = 0.0327\,m$ for a nodal deformation reaching $3.08\,m$). Reference solution is in green and U-Mesh prediction is in blue.

has a tremendous potential for problems requiring very fast simulations of objects undergoing interactions. Such interactions could be user-driven or the result of contacts with other structures.

Despite the promising results of our method, with the current U-Mesh, it is not possible to make very accurate predictions when applying a force somewhere out of the sampled input domain of the training data set. This limitation is also inherent to POD. In the same manner, we are restricted to the geometry used to train the network. This is due to the fact that neural networks are not good at extrapolating, hence the importance of a good sampling of the input domain when generating the training data sets.

Yet, there are several directions to investigate to make this approach more broadly usable. In particular in the context of biomechanics, the geometries are non regular structures, the inputs can be different from forces and the object characteristics (such as material parameters and boundary conditions) can vary from one patient to another. In next chapters we will see how to generalize the U-Mesh framework to patient-specific modeling of soft tissues.

Contributions of this chapter:

- The U-Mesh framework: nonlinear dimensionality reduction for displacement field estimation trained with controlled *finite-element* simulated data
- Predictions on two benchmark examples (cantilever beam and L-shaped structure) with very high accuracy in only $3\,ms$
- Constant computation time regardless of the size of the problem
- Publication: Mendizabal, A., Márquez-Neila, P., and Cotin, S., 2020. **Simulation of hyperelastic materials in real-time using deep learning. Medical Image Analyses.** Volume 59, 101569 10.1016/j.media.2019.101569

Part II

Patient-specific real-time simulations

Chapter 5
Patient-specific modeling

5.1 Introduction

In part I of this thesis work, we have explored how to simulate objects in real-time without taking care about the parameterization of the models. However, when considering *computer-assisted interventions*, the accuracy of the simulations and consequently their parameterization are essential. Depending on the surgical intervention, the demanded precision is variable but a *target registration error* of $5\,mm$ is usually accepted [Ruiter et al., 2006]. For instance, in image guided biopsies (where our mechanical model can be used to enrich the medical images), the tip of the puncturing needle must be in contact with the tumors whose size can be relatively small (less than $2\,cm$). To guarantee this level of precision the modelisation should be adapted to the patient anatomy and to its mechanical properties. As a matter of fact, the reliability of a biomechanical model highly depends on its parameterization. Therefore, once the constitutive law is chosen, the patient-specific material properties and boundary conditions must be carefully set in order to achieve the desired accuracy [Bosman et al., 2014].

Our *finite-element* simulations depend on parameters of the constitutive model such as Young's modulus, Poisson's ratio and the boundary conditions (BCs) (see Sections 2.3.1 and 2.3.2). Since soft tissues may be represented with an incompressible material, Poisson's ratio can be safely set to a value close to 0.5. However, the value of Young's modulus E is more difficult to estimate for a given organ as it varies with the age of the patient or even with treated pathology. For instance, considering hepatic surgeries, a cirrhotic liver is significantly stiffer than average livers. Therefore, values from the literature do not directly match each patient. Besides that, the location and the elastic properties of the attachments of the organ play a major role in the accurate approximation of the displacement field. Such BCs are not visible in the preoperative images and it is difficult to estimate them intraoperatively as they are often out of the field of view of the surgery.

The elastic properties of materials can be identified by solving inverse prob-

lems [Zhao et al., 2009, Gee et al., 2010, Lu et al., 2009, Sinkus et al., 2010] or using elastography techniques [Sarvazyan et al., 1998, Muthupillai et al., 1995, Xu et al., 2007] initially developed for diagnosis purposes. Some works have focused on the estimation of BCs intraoperatively such as [Peterlík et al., 2014], [Plantefeve et al., 2016] and [Johnsen et al., 2015] but these methods are difficult to use in practice as either additional intraoperative scanning is required or they are sensitive to anatomical variations. Moreover, when acquiring information intraoperatively, observational errors may occur, thus introducing uncertainty to the system. Alternative solutions accounting for such uncertainty rise from the use of Bayesian methods. For instance, authors in [Nikolaev et al., 2018, Peterlík et al., 2017], employed the *reduced-order unscented Kalman filter* (ROUKF) to model the BCs of a liver as stochastic parameters, leading to more accurate simulations of the deformations of the organ.

In this chapter we will briefly see how to obtain a patient-specific geometry of the patient's anatomy and then, similarly to works in [Nikolaev et al., 2018, Peterlík et al., 2017], we will describe a method to determine the patient-specific parameters of an organ using the ROUKF algorithm.

5.2 Patient-specific geometry

The first step towards patient-specific modeling is the patient-specific geometry of the organ. Generally, the 3D anatomical model of the organ is constructed from preoperative volumetric medical images such as *computerized tomography* (CT) scan or *magnetic resonance imaging* (MRI). Both acquisitions consist of a sequence of 2D slices that are merged into a 3D image using volume rendering. The color intensity of each voxel translates variation in material properties thus allowing to distinguish the different structures. Then, a process called *segmentation* is performed with a medical imaging software such as *3D Slicer*[1] or with deep learning algorithms [Zhou et al., 2019] in order to extract a volumetric surface of the organ and its internal structures (see Figure 5.1 for an example).

5.3 Model parameterization

We present here the procedure to estimate the value of Young's modulus and the BCs of an organ using observations of the target model using the ROUKF. To this end, each sought parameter p is described as a stochastic parameter associated to a Gaussian probability density function (PDF). Initially $p \sim \mathcal{N}(\mu_0, \sigma_0)$ with μ_0 the mean value of p reported in the literature and σ_0 its standard deviation. The aim

[1] www.slicer.org

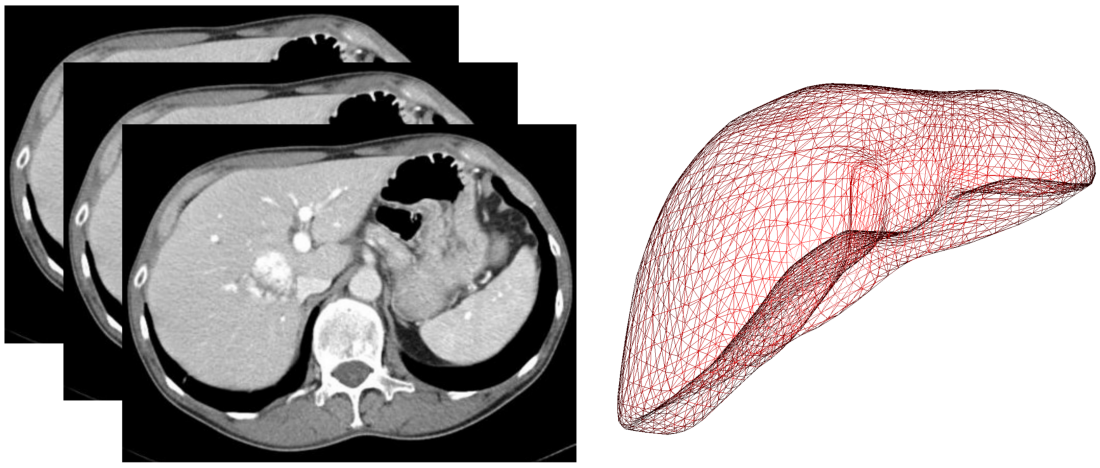


Figure 5.1: Sequence of 2D CT images of the liver and resulting surface mesh.

of the assimilation process is to iteratively reduce the standard deviation σ of p in order to find the most likely value for μ. To this end, the PDF of the parameter p is transformed based on observations. The transformation of the PDF is modelled using a ROUKF which can handle non-linear processes, and is computationally efficient [Moireau et al., 2011].

5.3.1 ROUKF: Overview of the algorithm

Once the FE model of the organ is built (from a preoperative CT scan for instance), the constraints imposed on the surface Γ_D need to be identified in order to generate a deformation. In other words, the organ attachments representing Dirichlet boundary conditions and the traction or displacement imposed on the free part of the boundary need to be identified. In this work, the former can be either fixed (Section 9.2.1) or set as a stochastic set of parameters (Section 9.2.2) and the latter can be determined intraoperatively. During the intervention, points in the surface of the organ can be tracked in each video frame. Such points are called *features* and are separated into *control features* and *observation features*. The control features govern the deformation of the liver model (imposed displacement on Γ_D or traction if a force sensor is available) and the observation features correspond to ground truth data (used in the filter *correction* phase to compute the Kalman gain). The control features can be selected close to the surgical tool and be used to prescribe displacements in the mechanical model.

An efficient implementation of a Bayesian inference method able to process nonlinear systems like our models is the *unscented Kalman filter* (UKF) [Julier et al., 1995]. Compared to an *extended Kalman Filter*, it does not require to compute the Jacobian of the system, which would be prohibitive given the size of our problem. The

unknown data to be estimated (the *stochastic state* of the system) is described as a Gaussian distribution, which transformation through the nonlinear system is performed using an *unscented transformation* (see [Julier et al., 1995] for details). The main idea is to parameterize the Gaussian distribution using a set of *sigma points*, which hold the mean and covariance information, but are easier to transfer through a nonlinear function. The general algorithm is described in Algorithm 2. It consists of a loop that contains two main steps. During the *prediction* step we form the new hypothesis about the estimated state, while during the *correction* step we correct it by comparing the predicted measurements with (noisy and partial) observations.

Algorithm 2: Main steps of unscented Kalman filter

1: **Initialize data:**
2: **set** $\mathbf{x}_1$ - *model positions and unknown parameters*
3: **set** $T = T(\mathbf{x}_1)$ - *finite element model*
4: **set** $\boldsymbol{\mathcal{I}}$, $\mathbf{P}_1$, $\mathbf{Q}$, $\mathbf{W}$ - *initial filter parameters*
5: **for** *each simulation step* i **do**
6: **Compute prediction phase:**
7: $\mathbf{x}_i^{\sigma_*} = \mathbf{x}_i + \sqrt{\mathbf{P}_i}\boldsymbol{\mathcal{I}}$ - *generate sigma points*
8: **for** *each sigma point* k **do**
9: $\widetilde{\mathbf{x}}_{i+1}^{\sigma_k} = T(\mathbf{x}_i^{\sigma_k})$ *get result from deformation step*
10: **end for**
11: $\widetilde{\mathbf{x}}_{i+1} = \mathcal{E}(\widetilde{\mathbf{x}}_{i+1}^{\sigma_*})$ - *compute predicted state as mean of sigma points*
12: $\widetilde{\mathbf{P}}_{i+1} = (\widetilde{\mathbf{x}}_{i+1}^{\sigma_*} - \widetilde{\mathbf{x}}_{i+1})(\widetilde{\mathbf{x}}_{i+1}^{\sigma_*} - \widetilde{\mathbf{x}}_{i+1})^T + \mathbf{Q}$ - *compute predicted covariance*
13: **Compute correction phase:**
14: **get** $\mathbf{q}_{i+1}^{(o)}$ - *observation features*
15: **for** *each sigma point* k **do**
16: $\widetilde{\mathbf{q}_{i+1}^{(o)}}^{\sigma_k} = H(\widetilde{\mathbf{x}}_{i+1}^{\sigma_k})$ - *get predicted observation*
17: **end for**
18: $\mathbf{P}_{\mathbf{xq}^{(o)}} = (\widetilde{\mathbf{x}}_{i+1}^{\sigma_*} - \widetilde{\mathbf{x}}_{i+1})(\widetilde{\mathbf{q}_{i+1}^{(o)}}^{\sigma_k} - \mathcal{E}(\widetilde{\mathbf{q}_{i+1}^{(o)}}^{\sigma_*}))^T$ - *compute cross covariance*
19: $\mathbf{P}_{\mathbf{q}^{(o)}} = (\widetilde{\mathbf{q}_{i+1}^{(o)}}^{\sigma_k} - \mathcal{E}(\widetilde{\mathbf{q}_{i+1}^{(o)}}^{\sigma_*}))(\widetilde{\mathbf{q}_{i+1}^{(o)}}^{\sigma_k} - \mathcal{E}(\widetilde{\mathbf{q}_{i+1}^{(o)}}^{\sigma_*}))^T + \mathbf{W}$ - *comp. obs. cov.*
20: $\boldsymbol{\mathcal{K}}_{i+1} = \mathbf{P}_{\mathbf{xq}^{(o)}}\mathbf{P}_{\mathbf{q}^{(o)}}^{-1}$ - *compute Kalman gain*
21: $\mathbf{x}_{i+1} = \widetilde{\mathbf{x}}_{i+1} + \boldsymbol{\mathcal{K}}_{i+1}(\mathbf{q}_{i+1}^{(o)} - \mathcal{E}(\widetilde{\mathbf{q}_{i+1}^{(o)}}^{\sigma_*}))$ - *compute corrected state*
22: $\mathbf{P}_{i+1} = \widetilde{\mathbf{P}}_{i+1} - \mathbf{P}_{\mathbf{xq}^{(o)}}\mathbf{P}_{\mathbf{q}^{(o)}}^{-1}\mathbf{P}_{\mathbf{xq}^{(o)}}^T$ - *compute corrected covariance*
23: **end for**

The prediction step can be very costly when using a model with many degrees of freedom, as it is the case when using a FEM method. Using the simplex method to generate the sigma points would require $m + 1$ simulations if m is the number of elements in the stochastic state vector (line 9 of the algorithm). With a mesh of n nodes and k stiffness parameters, this would mean $3n + k + 1$ simulations. A simple FEM mesh of only a few hundred nodes would be too time-consuming for a

clinical application, as it would require more than 300 simulations for each step of the assimilation process. To solve this issue, we use a ROUKF instead of the UKF. This method significantly reduces the computation cost since only $k+1$ simulations (in the best case) are required. This approach was proposed in [Peterlík et al., 2017].

Let us assume there are k unknown parameters in our model, so k different parameters to estimate that can be either the elasticity of the material or the elasticity of the organ attachments. Since we are using the simplex version of the ROUKF, there are $k+1$ sigma points meaning that $k+1$ evaluations of the model are performed in each prediction step of the assimilation process. At each step of the assimilation, the control features are extracted from the actual video frame and mapped onto the FE model through barycentric coordinates, in order to prescribe displacements. At the first step, μ and σ are initialized to μ_0 and σ_0 for each parameter. Then, $k+1$ vectors of parameters are sampled and $k+1$ simulations are performed. Each simulation corresponds to one of the sampled $k+1$ values of the parameter and they can be done in parallel as they are independent. After the simulations for all sigma points are performed, the *a priori* expected value and covariance matrix are updated. This is called the prediction phase. Later, in a correction phase, the extracted observation features are compared to the model predicted positions to compute the innovation that is used to compute the Kalman gain. The *a posteriori* expected values and the covariance matrix are computed based on the Kalman gain.

5.4 Conclusion

In this chapter we have seen the importance of the patient-specific parameterization of the *finite-element* models. The ROUKF allows for real-time estimation of the sought parameters based on intraoperative observations. In chapters 7, 8 and 9 we will see how to introduce this knowledge into the U-Mesh for allowing patient-specific predictions of the displacement field of an organ. In next chapter, we will see how this data assimilation process can help in robotic interventions.

Chapter 6
Force Estimation in Robotized Injections

6.1 Introduction and related work

The work presented in this chapter was produced at the beginning of this PhD. It consists of a thematically isolated example of combining *finite-element* simulations with neural networks in the context of robotized intravitreal injections.

Intravitreal injections are among the most common surgical interventions in ophthalmology with more than 4 million injections worldwide in 2014 alone [Ullrich et al., 2016]. This procedure is principally used in the treatment of diabetic maculopathy and for injecting vascular endothelial growth factor inhibitors in the treatment of age-related macular degeneration. Besides, we observe an increasing demand for such therapy due to the growing prevalence of diabetic patients and aging demographics. Intravitreal injections are mostly performed by doctors and the cost of such therapy has to be reduced. The increasing workload and the reduced reimbursement makes it difficult for hospitals to handle the situation. The time spent by the clinicians performing the injections has to be minimized while preserving the precision and the safety of the patient.

At the same time, robotic assistance in ophthalmology provides the ability to improve manipulation skills, along with shorter and safer surgeries [Meenink et al., 2012]. To this end, [Ullrich et al., 2016] proposed a robotized intravitreal device capable of assisting injections into the vitreous cavity. However, designing such robotic systems requires to solve multiple challenges in terms of safety, cost and time efficiency and usability. The position accuracy and the orientation of the needle are of particular interest since the injection must be performed in a small region (the pars plana) of the eye. If the region is missed, damage of the eye lens or the retina might occur. In addition to accurate positioning, the ability to estimate or measure the force exerted by the needle on the sclera during the procedure could offer an important additional safety for the patient [Jagtap et al., 2004].

As a matter of fact the knowledge of the force plays a central role in the control loop of surgical robots for patient safety during robotic assisted interventions [Weber et al., 2017]. The force can be either measured through force sensors or estimated through vision-based methods. The use of force-based control algorithms allows for an improved human-machine interaction, more realistic sensory feedback and telepresence. Beyond this, force sensing or force estimation can facilitate the deployment of essential safety features [Haidegger et al., 2017]. A considerable amount of work relying on force sensors has previously been done, focusing on the development of miniaturized devices to ease their integration with actual systems. Force sensors usually need to meet several additional requirements, such as being water resistant, sterilizable and insensitive to changes in temperature [Haouchine et al., 2018]. The major limitation of conventional sensors is thus the associated cost since most surgical tools are disposable [Haidegger et al., 2017]. To overcome this point, alternative solutions have been proposed, such as qualitative estimation of forces based on images [Mura et al., 2016, Haouchine et al., 2018].[Mura et al., 2016] introduced a vision-based haptic feedback system to assist the movement of an endoscopic device using 3D maps generated with a Shape-from-Shading method where the 3D shape of the surface is recovered from a 2D image of that surface. [Haouchine et al., 2018] incorporated the use of a biomechanical model of the organ in addition to the 3D maps to estimate force feedback in robot-assisted surgery. This approach is, however, limited to the ability to evaluate tissue properties of the organ quantitatively [Aviles et al., 2014].

Deep learning has already been suggested to improve existent characteristics of robotic assisted surgeries such as instrument segmentation and detection [Pakhomov et al., 2017], as well as force estimation. For instance, in [Aviles et al., 2014], interaction forces in minimally invasive surgeries are estimated with recurrent neural networks using camera acquisitions combined with kinematic variables and deformation mappings. In a follow-up paper, [Aviles et al., 2015] used a neuro-vision based approach for estimating applied forces in the same context. In [Aggarwal et al., 2018], authors used two neural networks to classify muscle force exertion levels to prevent musculoskeletal disorders based on features extracted from video data and blood volume changes. However, in all these approaches an intermediate step to determine the surface deformation is required.

In this chapter, we present a method for estimating contact forces directly from an OCT image of the scleral deformation without the need for a specific image feature extraction method beforehand, as in [Aviles et al., 2015, Aviles et al., 2014]. The technique relies on an image classifier for estimating force quantiles during robotized intravitreal injections. An imperative requirement for machine learning algorithms to work is the huge quantity of data to train on. Currently since intravitreal injec-

tions are executed manually, there is no available information on the force induced by the needle. Hence, we propose to build a patient-specific biomechanical model of the sclera to generate a very large number of force-OCT image pairs, which are then used for training any supervised machine learning method. We will show that this approach allows us to avoid the need for large data sets of real OCT images. The simulations are parameterized to match experimental results and compensate for the absence of real data. The parameterization can be carried out with different approaches. For instance, simple optimization algorithms could be used to minimize the errors between the output of the simulation and the OCT image over the set of possible parameters. However, in the view of an online estimation of the parameters, we choose to use a Reduced-Order Unscented Kalman filter [Moireau et al., 2011] to estimate the stiffness of the scleras. We then train a two-layer image classifier algorithm with the generated images and their corresponding forces. This solution allows a straightforward force estimation process to take place. This two-stage process makes it possible to classify force ranges with 93% accuracy while requiring very few experimental data, as demonstrated on several *ex vivo* porcine eyes undergoing robotically-controlled needle insertions.

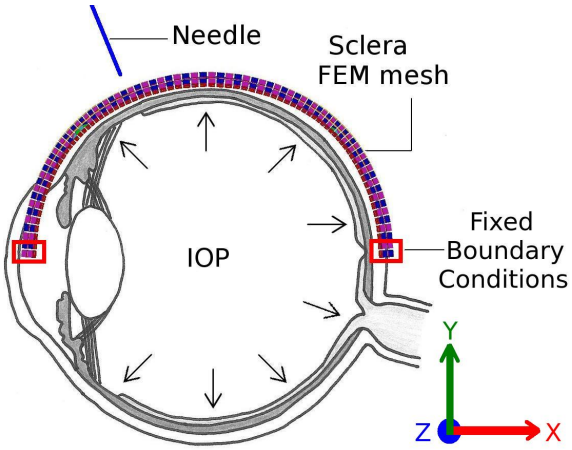
6.2 Method

In this work, we first construct a numerical model of the interesting portion of the eye to synthesize images of the deformed sclera under needle-induced forces. To be as close as possible to the actual organ behavior, we propose to use Bayesian inference as a means of identifying material parameters. Then, a data generation process takes place in order to train a neural network.

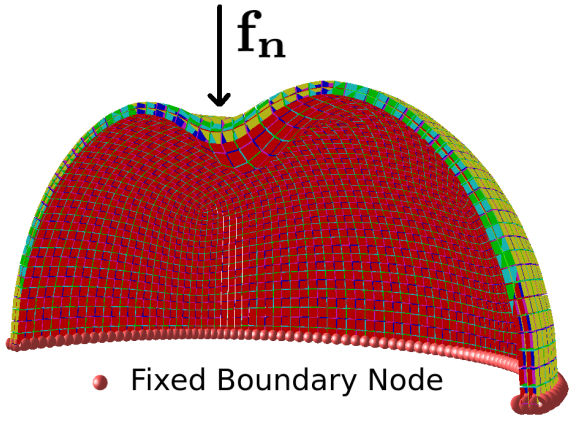
6.2.1 Training data generation

Biomechanical model We simulate the deformation of the sclera under needle-induced forces by modeling the eye as an elastic half-sphere subject to Dirichlet boundary conditions (see Figure 6.1a and Chapter 2 for details). Since applied forces and resulting deformations remain small, we choose to describe the stress-strain relationship as linear, using Hooke's law from Equation (2.19). Dirichlet boundary conditions are added to prevent rigid body motion of the sclera, while a constant pressure is applied to the inner domain boundary to simulate the intraocular pressure (IOP), as illustrated in Figure 6.1a. The intraocular pressure plays an important role in the apparent stiffness of the eye and its variability is well studied, as high eye pressure can be an indication for glaucoma. It is worth-mentioning that the common IOP measurement devices are influenced by the stiffness of the eye. The value measured by the tonometer is not absolute but linearly related to the Young's

modulus [Hamilton et al., 2008]. The external force due to IOP is given by $\mathbf{f_p} = S \times P$ where S is the surface area of the eye (in m^2) and P the intraocular pressure, given in Pa. Note that $\mathbf{f_p}$ is normal to the surface. To simulate the needle pushing on the sclera, we apply a local force $\mathbf{f_n}$ in a small region of interest near the virtual needle tip. The external forces of the system are then formed by the IOP, the force induced by the needle and the gravitational force $\mathbf{f_g}$ applied in the negative y direction.



(a) Simplified model of the sclera



(b) Deformed hexahedral mesh. Note that the colors of the hexahedra are just for visibility

Finite Element simulation We solve the equation for the constitutive model using the finite element method presented in Chapter 2. The eyeball is considered as nearly spherical and is discretized using a quadrilateral surface mesh of a half sphere of radius 12 mm, discretized using the Catmull-Clark subdivision method. The obtained quadrilateral surface is extruded conforming to the scleral thickness in order to generate almost regular hexahedra (see Figure 6.1b). Scleral thickness plays an important role in the deformation of the sclera. Under a given force, a thick sclera is less deformed than a thin one, therefore it is important to take this thickness into account.

In our simulation, we use a mesh composed of $14,643$ hexahedral elements, resulting in about 5 seconds of computation time to obtain the static solution of the deformation. All the experiments from this chapter are run on a Dell Precision laptop equipped with an intel Core i7 2.90GHz, a Quadro M1200 Graphics Processing Unit and 16Go of RAM. Since this computation has to be repeated thousands of times to generate the training data set, we take advantage of the linearity of the model and pre-compute the inverse of $\mathbf{K}$ to speed up the generation of the training data. This leads to a substantial computational speed-up (x10).

Model parameterization Our finite element simulations depend on parameters of the constitutive model and the geometry of the eye: Young's modulus, Poisson's ratio, intraocular pressure and scleral thickness. It is therefore essential to understand which parameters can be assumed constant, and which ones vary (and within what range) to properly generate training data.

Data from the literature [Olsen et al., 2002] report a thickness for corneal-scleral limbus in porcine eyes ranging from 630 μm to 1030 μm. We perform thickness measurements on porcine eyes and get a variation in thickness of more than 35%. Hence, we choose to consider the thickness as a parameter in the training, and we simulated scleras with five different thicknesses: 400 μm, 500 μm, 600 μm, 700 μm and 800 μm.

The IOP is also known to vary from patient to patient, and can be measured using a tonometer. So if we were to apply our method on a patient, we could use this information to parameterize the simulations used for the training. However, this is different for the porcine eyes used in our study. The IOP decreases gradually with postmortem time [Balci et al., 2010] and is halved only three hours after death. The IOP we measured ranged from 1 to 4 $mmHg$ which is very small and does not affect much the deformation of the sclera. Therefore we considered the IOP constant and equal to 2 $mmHg$ (266 Pa) for our experiments and simulations.

The Poisson's ratio $\boldsymbol{\nu}$ is a constant value for the sclera according to the literature [Asejczyk-Widlicka et al., 2008] and can be set to $\boldsymbol{\nu} = 0.45$. This leads to a nearly incompressible behavior during the deformation. With this in mind, we can reasonably assume that our training data set can be generated from numerical simulations in which the thickness, Poisson's ratio and IOP are known and constant.

The value of Young's modulus E, however, is more difficult to estimate as it varies depending on the porcine breed and experimental conditions. Therefore, values from the literature are not directly applicable. On the other hand, measuring it using a "classical" experimental biomechanics approach would be burdensome. For this reason, we propose to use a Bayesian approach to estimate the value of E using observations from our OCT images. Such an approach could also be used to perform data assimilation on actual patients.

The knowledge of the elasticity parameters is key to build a good model of the sclera. The goal of this work being the estimation of the force range based on the shape of the sclera, it is therefore very important to correctly estimate E to avoid introducing errors in the force prediction. Since the exact value of E is not *a priori* known (only its average value based on data from the literature), we describe it as a stochastic parameter associated to a Gaussian probability density function (see Section 5.3 in Chapter 2).

Data set generation Once the model is correctly parameterized, we generate a synthetic data set $\{(\mathbf{f}_{\mathbf{n}k}, I_k)\}_{k=1}^{N}$ of N samples where $\mathbf{f}_{\mathbf{n}k}$ is the needle-induced force for sample k and I_k is the corresponding simulated image of the deformation. The forces range from 0.0 N to 0.06 N and are applied at random locations and normally to the scleral surface for each of the different thicknesses. For each sample, a 2D cross-section of the complete 3D mesh is extracted in order to simulate the OCT image I_k. The simulated images have to be informative of the scleral thickness and its deformation, hence a binary image suffices for our purpose. Therefore the images are post-processed with a contour detection algorithm (*findContours* OpenCV function) and binarized (*threshold* function).

6.2.2 Neural network for image classification

We look into modelling a function that can estimate the range of forces applied by a needle when observing a single OCT image and the deformation within it. To do this, we make use of an artificial neural network (NN) to provide a robust and reliable function capable of estimating forces applied to the sclera (see Chapter 3 for details). In our case, the input to the model is the cross-sectional OCT image that depicts anatomical information of the sclera, from its surface to 1 millimeter below. In order to provide high-frame rate imaging, we opt to use 2D OCT cross-sections as the imaging modality over 3D volumetric OCT scans that are more common but slower to acquire. In our set up, we use the low-cost solution introduced in [Apostolopoulos et al., 2017] to image 2D OCT cross-sections.

While we are interested in estimating the force applied by the needle, we choose to categorize the applied forces into three interval ranges. This reduces the need to be sensitive to exact force values, which not only is unnecessary in this instance, but also allows us to set up our inference task as a classification problem, whereby forces are grouped into ranges of clinical relevance.

To do this, our NN consists of two fully connected hidden layers and a classifier as output layer (see Figure 6.2). The input layer has 5600 neurons (corresponding to the size of the input images 140×40) while the hidden layers have 600 neurons. These optimal values are found through a grid search. We use ReLu activations throughout the network and a softmax activation at the output layer. The network was implemented using Tensorflow and the Keras Python library.

Given the above simulation model, we can train our NN with virtually an *infinite* amount of synthetically generated data. With the objective of stopping an incumbent needle from damaging the sclera, we use our simulation model to generate OCT images with forces between 0.0 N and 0.06 N applied on the Finite Element mesh of the sclera. This range was then divided into three ranges, or classes, consisting of:

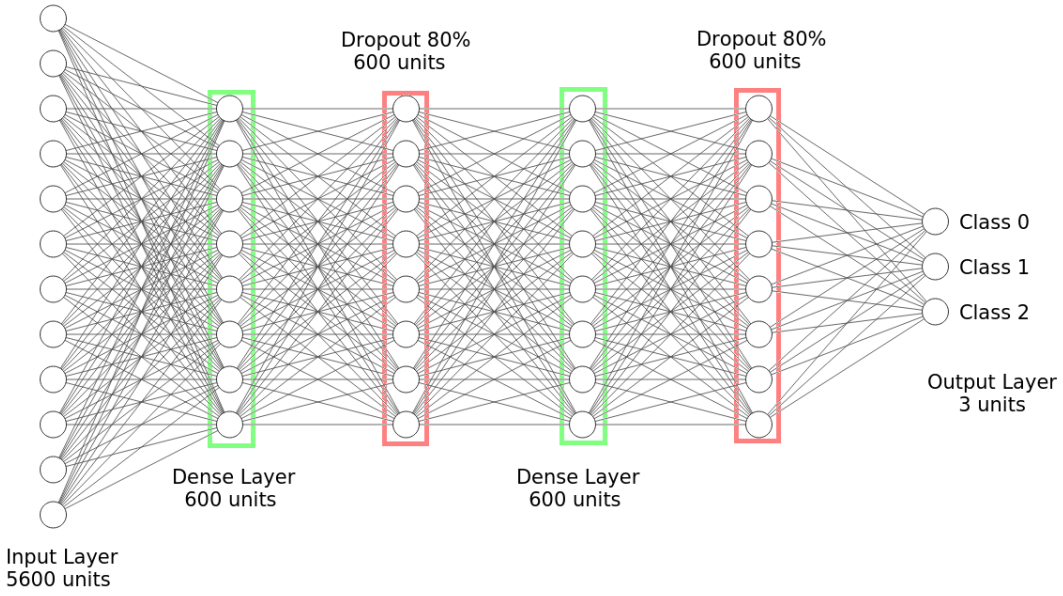


Figure 6.2: Artificial neural network used for image classification. The input is an image of size 140×40 pixels resulting in an array of size 5600.

- Class 0: force values smaller than 0.005 N
- Class 1: force values from 0.005 N to 0.03 N
- Class 2: force values bigger than 0.03 N

whereby class 0 corresponds to virtually no danger to the sclera, class 1 indicates a considerable force and class 2 is a dangerous force that should trigger a needle removal signal. The complexity of our problem makes it challenging to establish a clear cut between "no danger to the sclera" and "a needle removal is advised". For this reason we introduced an intermediate class which can be seen as a gray area between the two bounding classes. If the force is classified in this gray area, it becomes the clinician's responsibility to stop or continue the insertion.

We train our NN from scratch using a gradient descent and the cross-entropy loss. A 0.8 dropout factor between layers was used to help with generalization.

6.3 Experimental results

In this section, we first describe how data acquisition on *ex vivo* porcine eyes was performed. Then, we present the virtually generated data set, and associated training, with an emphasis on the parameterization of the biomechanical model. Finally,

we validate the neural network on unseen real data to demonstrate the high level of accuracy of our approach at classifying contact forces.

6.3.1 Experimental set up

To validate the predictive accuracy of the NN on real data, we acquired a collection of M samples $\{(\mathbf{f}_{\mathbf{n}k}, I_k)\}_{k=1}^{M}$ from *ex vivo* porcine eyes. Five porcine eyes were obtained from the local abattoir and transported to the experiment room while kept at low temperature. Experiments began within 3 hours of death, and we ensured they were completed less than 10 hours postmortem. During the tests, the eyes were moisturized with water and fixed with super glue on a 3D-printed holder to ensure fixed boundary conditions on the lower half of the eyeball. We choose super glue as it preserves the tightness of the eyeball. We then measured the IOP with a tonometer and obtained, for all the eyes, an IOP close to 2 $mmHg$ (about 266 Pa). The intraoperative pressure is lower than usual since it decreases dramatically after death [Balci et al., 2010]. However, for the results presented here, we did not inject any fluid during the experiments to compensate for this low IOP.

A surgical robotic arm that was designed for high-precision drilling during robotic cochlear implantation [Weber et al., 2017] was used to measure the force applied by a needle. The robotic arm is fitted with a six-axis force/torque sensor (Mini40, ATI) at its wrist underneath a quick release for the end effector. After mounting a 22G Fine-Ject needle (0.7 x 30 mm) at the tip of the end effector, the robotic arm guided the needle forward along its axis. The needle was positioned as close as possible to the B-scan without intersecting it to bypass the generation of shadows in the OCT image. The margin between the needle and the B-scan was considered in the simulation. The robot was programmed to move towards the center of the eye along a path normal to the sclera. Contact and insertion forces, in the direction of motion, were continuously recorded during the insertion process. We also recorded the associated OCT images by storing B-scans over time. The custom-designed OCT device uses a 840 $nm \pm 40$ nm wavelength light source, with an A-scan rate of 50 kHz for a resulting 2D image of resolution 512×512 pixels with 12 bits per pixel, corresponding to an area of 15×4 mm^2. In these images, the black bands surrounding the OCT focus are removed leading to images of size 140×40 pixels.

In Figure 6.3, a fragment of the collected data including 5 complete acquisitions is shown. To each trial corresponds one OCT image captured at a punctual time and a continuous flow of force values. The forces are filtered so that the noise is reduced. As the images were collected at a lower frequency than the forces, the corresponding forces were averaged over an interval of two seconds around the imaging time. The vertical lines correspond to the imaging times. The relaxation properties of soft tissue explain the slight decrease of the force a few seconds after the force is applied.

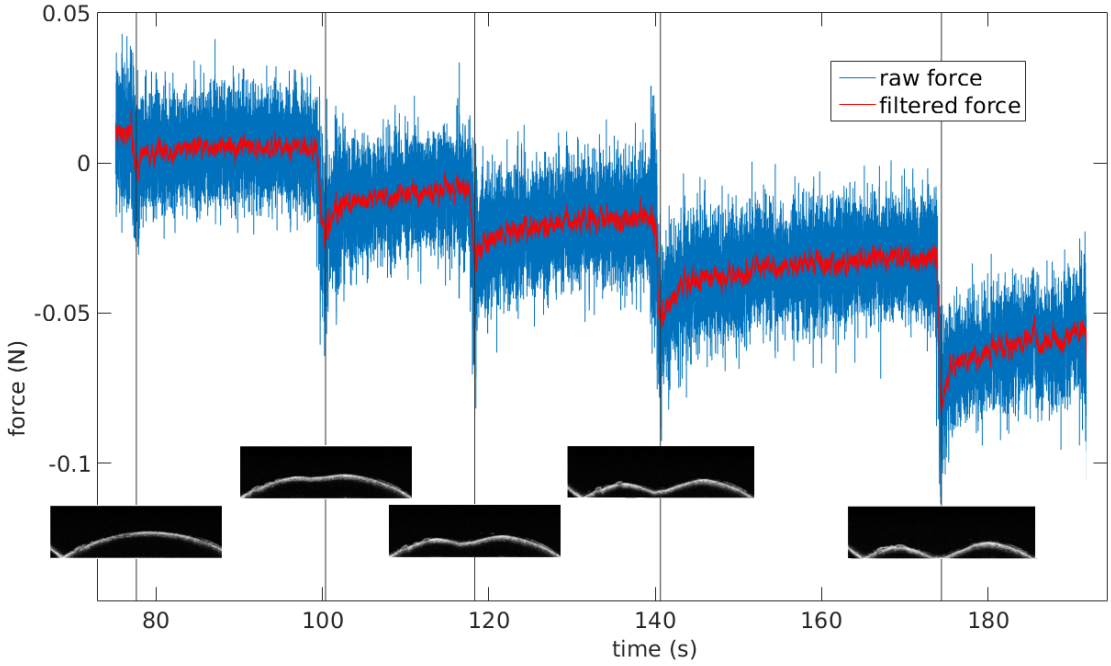


Figure 6.3: Continuous acquisition of the robot force sensor and corresponding punctual OCT images for one eye. The vertical lines give the imaging times.

eye ID	1	2	3	4	5	Total
scleral thickness in μm	800	500	650	560	800	-
number of samples in class 0	0	1	2	2	2	7
number of samples in class 1	11	2	0	1	1	15
number of samples in class 2	14	2	3	2	2	23
Total	25	5	5	5	5	45

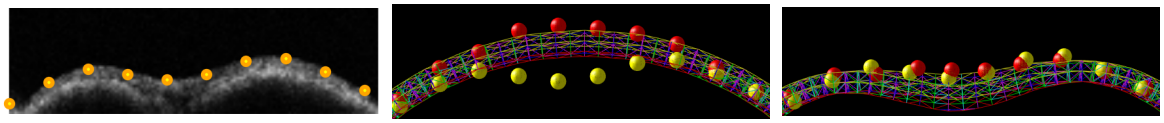
Table 6.1: Distribution of samples in the three force ranges and scleral thickness for each eye.

The computed average force is used as ground truth for the following supervised learning. For each eye, one position near the corneal-scleral limbus was selected and the needle moves forward 0.5 or 1 mm in the same direction. Note that the pierced samples are excluded from the data set. Overall, 45 valid trials were performed on different eyes. The spread of each acquisition among the force classes is depicted in Table 6.1.

6.3.2 Identification of the Young's modulus

According to the collected porcine data, the Young's modulus E of the biomechanical model is estimated using the ROUKF. The PDE of E is transformed based on observations taken on the OCT images acquired during the experiment on an eye

(see Figure 6.4a). We consider the corresponding ground truth force (measured with the robot's force sensor) to run the simulations.



(a) (b) (c)

Figure 6.4: (a) Observation points in the real OCT image. (b) Predicted observations (in red) and real observations (in yellow) for the first estimation of E. (c) Predicted observations (in red) and real observations (in yellow) for the final estimation of E.

According to [Asejczyk-Widlicka et al., 2008], we set μ_0 and σ_0 to 0.49 MPa and 0.34 MPa respectively. We estimated the Young's modulus for each eye independently based on all OCT images where observation points can be extracted. We extract 10 observations on each OCT image for a given eye (see Figure 6.4a). The finite element mesh is registered to the OCT image using rigid registration. To each observation in the OCT (yellow dots) corresponds a predicted observation attached to the model (red dots). During the assimilation, the known force is applied and the observations obtained are used by the filter correction phase to compute the Kalman gain. We employ the simplex version of the ROUKF, and since we only want to estimate one parameter, two evaluations of the model are performed in each prediction phase of the assimilation process (i.e. there are two sigma points). The evolution of μ and σ with the iterations of the ROUKF for eye 1 are shown in Figure 6.5. After convergence, we see that the final value of μ is 0.27 MPa, as reported in first column of Table 6.2. When applied to other porcine eyes, estimated values of E vary only slightly, as seen in Table 6.2. Therefore we consider the Young's modulus E to be constant across all our experiments, with an average value of 0.25 MPa. Using this values leads to very good visual agreement between the OCT images and the simulations for the three force ranges for all the eyes (see Figures 6.6 and 6.7) thus validating the assumption of linearity.

	eye 1	eye 2	eye 3	eye 4
μ in MPa	0.27	0.22	0.24	0.29
σ in MPa	$4.7e^{-3}$	$6.7e^{-3}$	$3.9e^{-3}$	$4.9e^{-3}$

Table 6.2: Mean and standard deviation of the estimated Young's modulus in 4 different porcine eyes after performing a data assimilation with a ROUKF. The fifth eye is not included since the observations required in the assimilation process are not consistent across the OCT acquisitions for this eye.

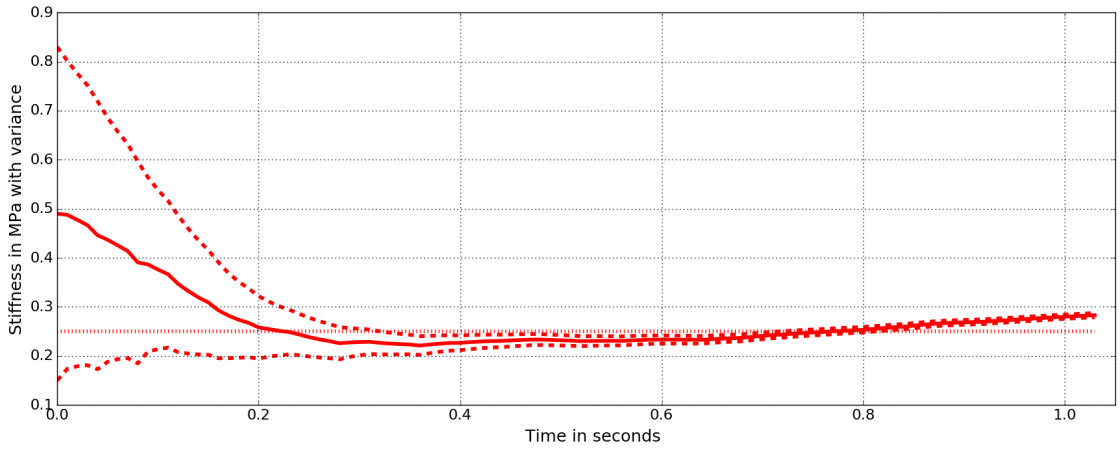


Figure 6.5: Variation of μ and σ for the Young's modulus estimation using the ROUKF for 24 OCT images. The value of the parameter converges to 0.27 MPa.

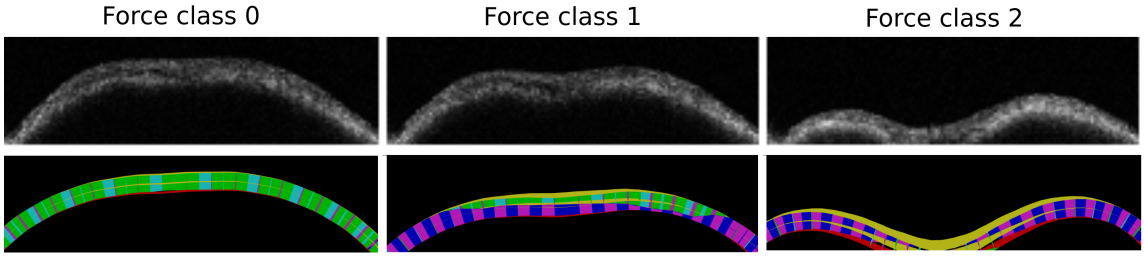


Figure 6.6: Real OCT images of the eye used for the data assimilation and matching simulated images with $E = 0.25$ MPa for the three force ranges.

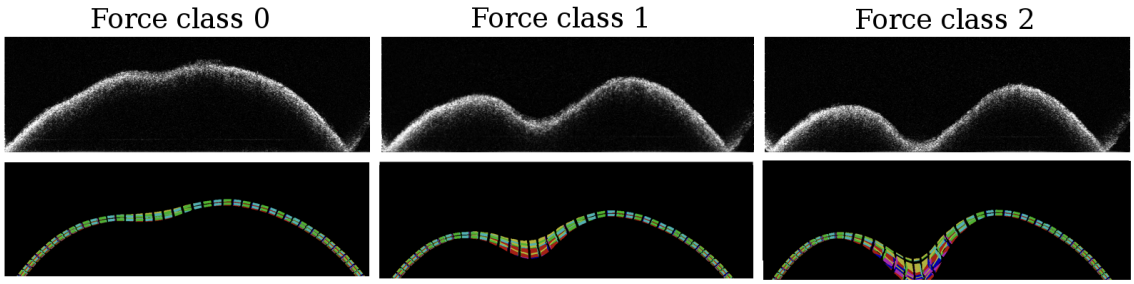


Figure 6.7: Real OCT images of the sclera and their matching simulated images with $E = 0.25\ MPa$ for the three different force ranges.

6.3.3 Training data generation

Data set generation To run our simulations we used the Simulation Open Framework Architecture[1]. We generated 3200 images of deformed scleras undergoing normal forces going from 0.0 N to 0.045 N for the smallest thickness. For each of the other four thicknesses, we generated 4000 images of deformed scleras where the forces vary from 0.0 N to 0.06 N at different random locations. For each thickness, the simulation took approximately half an hour. In Figure 6.7 are shown different examples of the output of the simulation (bottom) matching the OCT images (top). Overall, we created a data set having 19200 synthetic images within 2 hours and a half (see Figure 6.8b). In Figure 6.7 below we show some synthetic images.

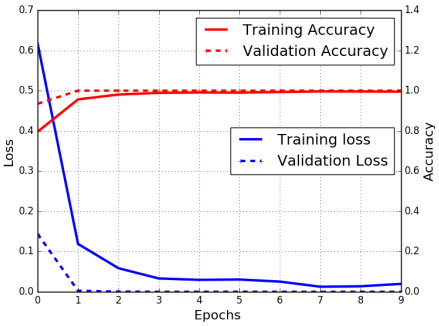
Neural network training The generated labelled data set is split such that 90% of the images is used for training and the remaining 10% is left for validation. All in all, 17280 images are used to train the artificial neural network that is validated on the other 1920 images. Figure 6.8a depicts the accuracy and the loss of the neural network on both training and validation data sets over each epoch. The validation accuracy curve displays 100% accuracy when classifying unseen synthetic images. This curve is above the training accuracy curve probably because of the high dropout applied during the training.

6.3.4 Validation on unseen real data

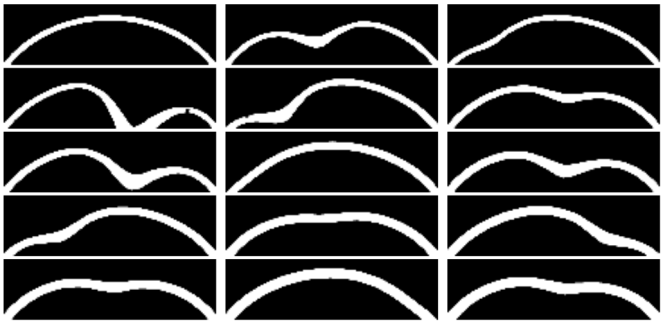
Our work aims at classifying force levels using only OCT data of the deformed scleras as input. All the OCT images collected during the experiments are filtered to obtain inputs similar to the synthetic ones (see Figures 10.2, 6.9b and 6.8b).

Once the OCT images have been processed to look like the simulated ones, we can use them as input to the NN and perform force predictions. For each OCT acquisition, the force measured by the robot is converted into a class label (0, 1 or 2) and is

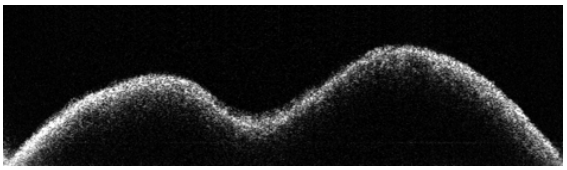
[1]https://www.sofa-framework.org/



(a) Loss and Accuracy curves for training and validation sets



(b) Fragment of the training data set generated by our numerical simulation



(a) Unprocessed OCT image

(b) Filtered OCT image

taken as the ground truth (target class). A class with label 0 corresponds to a minimal force for which there is no risk of damaging the sclera. A class with label 1 corresponds to forces ranging from $0.005N$ to $0.03N$ and indicates that the sclera is being considerably deformed. A force class with label 2 means that the sclera is potentially being damaged and that a withdrawal of the needle is advised.

The performance of the classification is reported in the confusion matrix in Table 6.3. Each row of the table gives the instances in a target class, and each column gives the instances in a predicted class. For each category, we highlighted the correct decisions in red and show that the overall accuracy of the classifier is very high (93%). For all the experimental data set, the lowest score of the NN was obtained for the force class 0 with 71% accuracy. For target class 2, the precision reached 100%. In Figure 6.10 are shown the raw force values (measured with the robot's force sensor) and the force range thresholds. The plot shows that the forces measured are uniformly spread throughout the force ranges. Note that the three misclassified samples are close to the upper bound of the force range and they are overestimated. We believe that misclassification of contact forces in the lowest ranges has a limited impact since the risk of damaging the sclera with such forces is almost null. On the other hand, it is essential that the forces of range 2 are predicted correctly, which appears to be the case on our (limited) data set.

Target \ Prediction	Class 0	Class 1	Class 2	Precision
Class 0	5	2	0	0.71
Class 1	0	14	1	0.93
Class 2	0	0	23	1.00
Recall	1.00	0.88	0.96	

Table 6.3: Confusion matrix

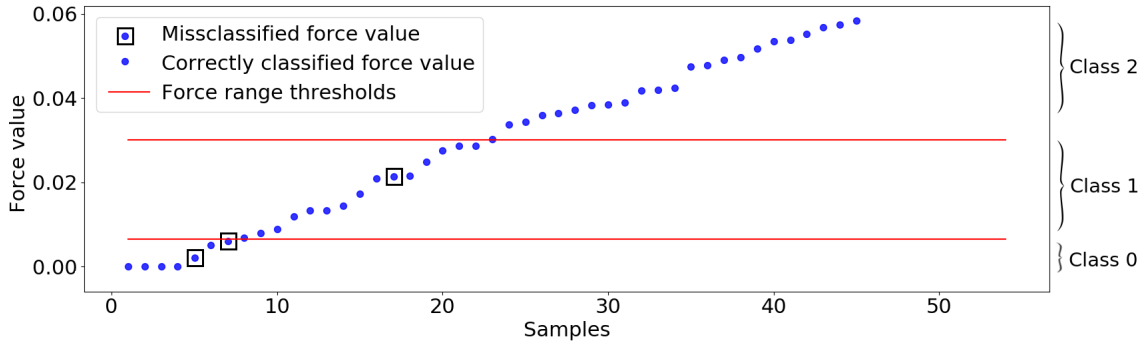


Figure 6.10: Filtered force values corresponding to the 45 OCT acquisitions. The three misclassified samples are overestimated. Note that we only kept the samples under $0.06N$ which is the point of rupture of the thinnest scleras. To correctly account for the elements above that point of rupture, the puncturing process should be included in our model which would complicate the simulation without bringing any gain.

6.4 Conclusion

This chapter introduces a method for improving the safety features of robotized intravitreal injections. We show that our vision-based method, which combines numerical simulation and neural networks, can accurately predict the level of force applied by a needle, using only 2D OCT images of the scleral deformation. By being real-time, this classification can lead to an immediate withdrawal of the needle once it reaches a certain alarm threshold. To cope with the issue of performing a large number of experiments to populate our training data sets, the NN is trained on synthetic images generated from simulations of sclera deformations. To automatically parameterize the simulations from the experimental data, we use a Kalman filter which performs data assimilation using the sequence of OCT images.

This work was carried out at the beginning of the PhD. At that time, we were inexperienced in deep learning algorithms so we selected the simplest network architecture for solving our classification problem. With the current experience, if we were to do this work again we would probably use a CNN instead. Indeed, CNNs are

aimed to deal with images and they are usually faster to train than fully connected neural networks (due to the smaller number of learnable parameters). Moreover, CNNs are less sensitive to image framing and scaling and would be better suited if we were to add real OCT images in our data sets.

Moreover, in order to use our method in clinical practice (e.g. *in vivo* eyes) further simulation efforts should be performed. For instance, anatomically reasonable boundary conditions should be considered such as muscle fixation and eye movement should equally be included. It also seems essential to include the intraocular pressure as an additional input in the data set. Indeed, IOP is known to play a role in the deformation of the sclera and cornea of patients. To perform the prediction, a measure of the IOP would simply need to be performed on the patient using a tonometer. The reason we have not done it yet comes from the inaccuracy of the IOP measurement which is influenced by the stiffness of the eye. This coupling makes it more challenging to parameterize our simulations. Another improvement of the method would be to use adaptive force thresholds depending on the scleral thickness and stiffness as for thinner or softer scleras, the puncture force is lower. The simulated images could also be improved to match actual surgical scenarios better. In particular, adding the shadows generated by the needle in the OCT image would be an essential feature. We also propose to address the sensitivity of the NN predictions to image framing and scaling by randomly cropping each simulated image, and augmenting the data set with these additional images. Note that in our data generation process, the optical interference phenomenon happening in OCT is not simulated. Instead, the synthetic and real images are binarized. This skeletonization is a way of filtering the complicated aspects of the OCT images that are *a priori* non-informative for our purpose. Training a network directly from the OCT images without the skeletonization would lead to a significantly more complex problem requiring the simulation of light propagation and the use of Convolutional Neural Networks.

Finally, the objective of this work was to estimate a force range, but it would also be pertinent to estimate the location and the angle of the force as a slight error in the latter might damage the retina or the eye lens. In this context we could augment the data set by including various needle insertion angles and locations, and use a regressor to predict the complete force vector.

Contributions of this chapter:

- Simple fully connected neural network for force classification
- Image-based force estimation in robotized procedures (no force sensor needed)
- Training data generated with a simplified *finite-element* model of the eye
- Identification of the stiffness of the sclera using Kalman filters
- 93% accuracy in the force classification on real porcine data
- Publications:
  - Mendizabal, A., Fountoukidou, T., Hermann, J., Sznitman, R., Cotin, S., 2018. **A Combined Simulation and Machine Learning Approach for Image-Based Force Classification During Robotized Intravitreal Injections.** In International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 12-20.
  - Mendizabal, A., Sznitman, R., Cotin, S., 2019. **Force classification during robotic interventions through simulation-trained neural networks.** International journal of computer assisted radiology and surgery, 14(9), pp. 1601-1610.

Chapter 7

The U-Mesh for Augmented Hepatic Surgery

7.1 Introduction

In Chapter 4, we have presented the performance of the U-Mesh framework on simple geometries, matching the required grid-like structure for the input to the network. In this chapter, we will see how to extend the method to any kind of geometry. In particular, we will first show the ability of the U-Mesh in predicting the deformations of a liver geometry. Second, we will apply it to augmented hepatic surgery by registering the preoperative images to the intraoperative ones.

In liver surgery, it is essential to locate the internal structures (such as tumors and blood vessels) that need to be preserved for the post-operative regeneration of the liver tissue. While the initial position of these structures is known from the preoperative images, their actual position during surgery is often hidden or uncertain. To guide the surgeon, augmented reality techniques are used to enrich visual information through fusion of intraoperative images and a preoperative 3D model of the patient's anatomy. This is usually done by overlaying a virtual representation of the liver built from preoperative images over intraoperative images or through augmented reality glasses. However, surgical manipulations and interactions with the surrounding anatomy can induce significant deformations to the patient's liver. As a consequence, the virtual model of the liver has to account for non-rigid transformations and produce its deformed state in real-time, which is difficult given the complexity of the physical systems needed for accurate biomechanical modeling. Therefore, in order to build an augmented view of a liver during surgery we need to perform an elastic registration of the preoperative model to the intraoperative images acquired with a 3D imaging device (see Figure 7.1). While in *minimally-invasive surgeries* a laparoscopic camera can be used to acquire a video of the abdominal cavity, in open surgeries an RGB-D sensor can capture the surface deformation of the tissues. From such images, a partial point cloud of the liver surface can be extracted using one of the methods listed

in [Petit et al., 2018]. In this chapter we will see how the U-Mesh can produce an estimation of the complete displacement field given such a point cloud.

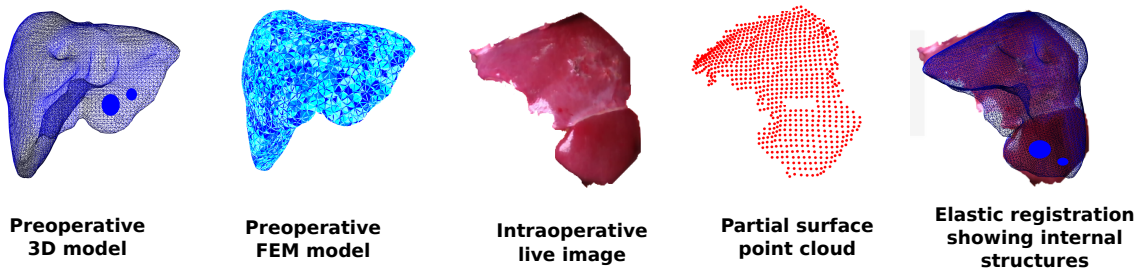


Figure 7.1: Augmented reality pipeline: preoperative internal structures are mapped in real-time onto the live image of the organ using a FEM model.

This chapter is divided in two main segments. To start with, the U-Net learns to predict the displacement field of a virtual liver given an input contact force. In a follow-up, the U-Net is used in an augmented reality context, where the full volumetric displacement field needs to be estimated from a partial surface deformation. In both cases the network is trained with FEM-generated data and the procedure is very similar to the one presented in Chapter 4 except from the meshing procedure.

7.2 Displacement field on regular grids

To train our network, we need to generate many samples consisting of a volumetric output displacement field $\mathbf{U_v}$, given an input constraint $\mathbf{C}$ (contact force or surface displacement). The simulation that generates this data needs to be as accurate as possible, yet computationally efficient to make it possible to generate data and train the network in less than a day (i.e. in the shortest amount of time available between preoperative data acquisition and surgery). As seen in Section 4.2, the input to the U-Net must be a regular grid. In the following we will present two options to fulfill this requirement.

First, the nodes of the *finite-element* mesh can be mapped into the nodes of the input grid. That way, the FE mesh and the U-Net input grid are independent and any type of elements can be used to mesh the object (tetrahedral elements, non regular hexahedral elements, linear or quadratic elements etc). In order to mesh complex geometries (in particular to handle the boundaries of an object), using 4-node tetrahedral elements can be more convenient than using 8-node hexahedral elements [Wang et al., 2004]. Therefore we propose to start discretizing a liver geometry with tetrahedral elements and map a regular grid onto the tetrahedral mesh. The regular grid follows the deformation of the FEM mesh (see Figure 7.4b). Only the nodes

of the grid that are inside the liver volume are mapped and the outer nodes are zero-valued. Indeed, the mapping is only exact for inner grid nodes since we can use the FE shape functions which is not the case when solving the grid nodal values outside the geometry (ill-posed problem). The objective being to find the location of the internal structures, this simplification is not a limitation.

The second option consists of building the FE mesh such that it directly matches the U-Net grid. Here we will rely on 8-node hexahedral elements known for their better convergence, and lock-free behavior, especially with close to incompressible materials (such as the liver) and strong shear stresses [Benzley et al., 1995]. In addition, we combine this choice of elements with an *Immersed-boundary method* (IBM), which allows us to create an hexahedral mesh directly from the segmented CT or MRI image (see Figure 7.2). The surface mesh is embedded into a sparse hexahedral grid (Figure 7.5a), that is in turn embedded into a regular grid (Figure 7.5b) which matches the structure of the first layer of our U-Net. The sparse grid consists of rectangular cuboid cells. Cells that are overlapping the domain boundary are kept, therefore approximating the exact shape and volume of the object. The choice of

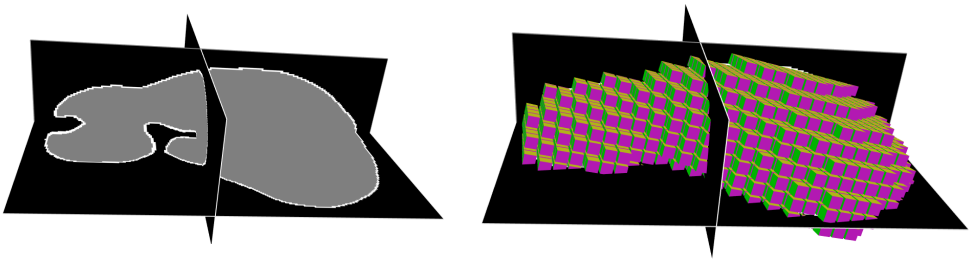


Figure 7.2: Example of sparse grid discretization generated from a preoperative CT. Combined with an Immersed boundary method, it allows the use of regular hexahedral meshes to simulate the deformation of the organ, and can be used with our CNN.

discretizing the initial domain with a sparse and regular grid brings several benefits. The usual trilinear interpolation functions of an 8-nodes hexahedral element are reduced to a linear mapping, and, similar to 4-nodes tetrahedral elements, its jacobian remains constant inside the element. However, using such discretization requires particular care of the boundary elements. Volume integration of the displacement field inside these partially filled cells is carried on by recursively subdividing the cell into 8 sub-cells. The stiffness matrix inside a boundary cell is then accumulated from its sub-cells using the linear mapping, as in [Düster et al., 2008]. Since we are using a fine mesh of the domain, only one level of subdivision is enough to obtain an accurate approximation of the volume integral. Note that the finer the grid, the

smaller is the difference between the exact volume and the one represented by the sparse grid.

7.3 The U-Mesh applied to a liver

7.3.1 Results on a synthetic liver

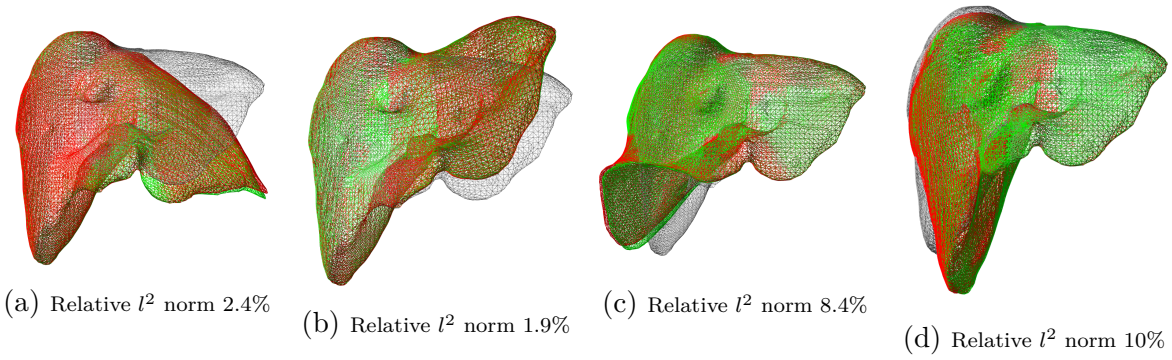
A surface mesh is obtained from a preoperative CT scan of a human liver. The length of the liver is $0.2\,m$. We will first mesh the liver using T4 elements and then using H8 elements. In both cases the Young's modulus will be set to $5,000\,Pa$ and the Poisson's ratio to 0.48.

Liver meshed with tetrahedral elements In this paragraph the liver geometry is discretized into 4859 tetrahedral elements (1059 nodes in total). A $16 \times 15 \times 16$ regular grid is mapped onto the tetrahedral mesh and follows the deformation of the FEM mesh (see Figure 7.4b). Dirichlet boundary conditions are then added by fixing some nodes in the area separating the two lobes to mimic the effect of the vascular tree and of the falciform ligament of the liver [Abdel-Misih and Bloomston, 2010]. For the training data generation, normal forces of random magnitudes are computed on the liver surface and mapped to the grid nodes using barycentric coordinates. Only one force is applied at each time step on a small region of the surface. We decided to limit the size of the data set to fit the time requirements of a clinical routine where sometimes only a few hours are available between the preoperative CT scans and the surgery. Hence a data set of only $2,000$ samples is generated in $135\,min$. $N = 1,600$ samples are used to train the network in $149\,min$ and $M = 400$ samples are left for validation.

Once the network is trained, we make predictions over the validation set. The results are reported on Table 7.1. The sample with maximal error is shown in Figure 7.4a. The outputs are predicted in only $3\,ms$. In Figure 7.3 are shown some samples of U-Mesh-deformed livers and their corresponding relative errors computed at one of the lobe tips. The output of U-Mesh is in green whereas the reference solution is in red.

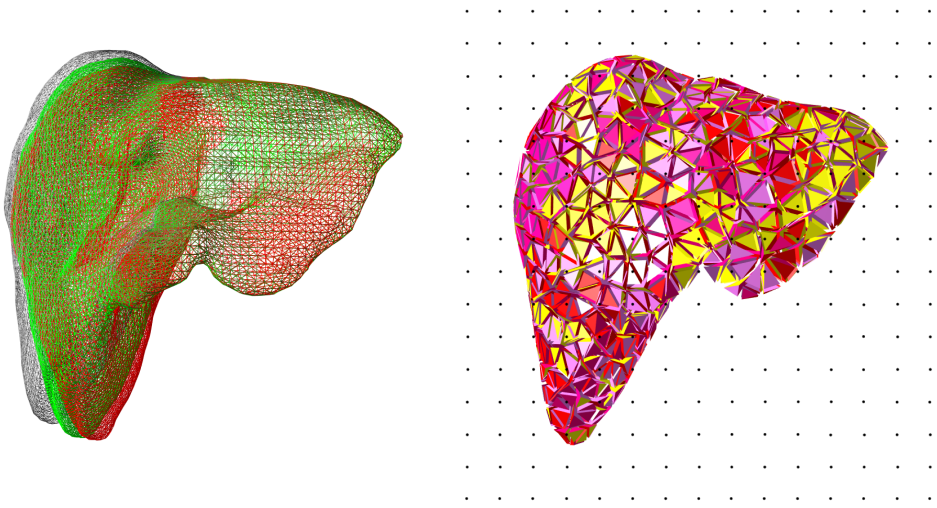
c	k	FSS	$\overline{e}$ in m	$\sigma(e)$ in m	pred_t in ms	train_t in min
128	3	1024	5.33e-05	6.03e-05	3	149

Table 7.1: Error measures on a liver of length $0.2\,m$ discretized with tetrahedral elements. The maximal error in the testing data set is equal to $4.9e-04\,m$ (sample in Figure 7.4a) for a maximal deformation over the testing data set of $0.088\,m$.



(a) Relative l^2 norm 2.4% (b) Relative l^2 norm 1.9% (c) Relative l^2 norm 8.4% (d) Relative l^2 norm 10%

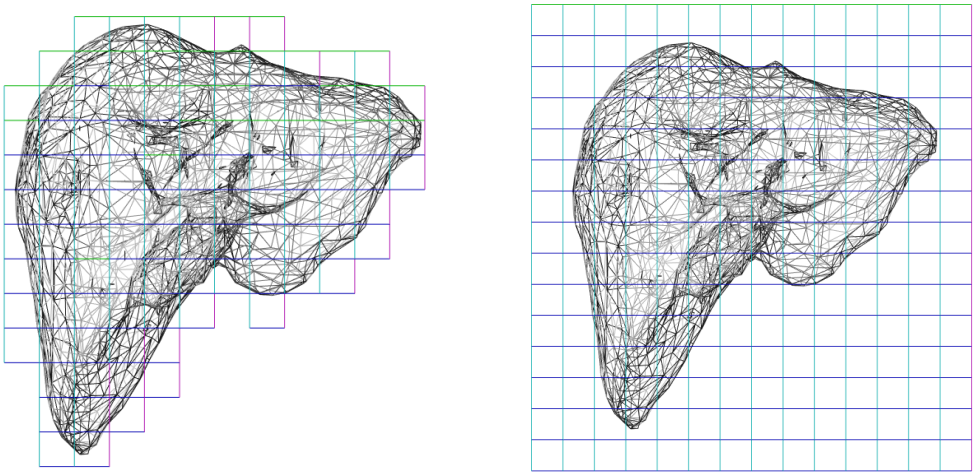
Figure 7.3: Various liver samples from the testing data set and corresponding relative errors computed on the tip of the deformed lobe for the tetrahedral topology. The rest shape of the liver is shown in grey.



(a) (b)

Figure 7.4: (a) Maximal average nodal error. (b) Regular grid mapped onto a FE tetrahedral mesh.

Liver simulation using hexahedral grids This time the surface mesh is Immersed in a $16 \times 15 \times 16$ grid (see Figure 7.5b) resulting in 732 H8 elements (see Figure 7.5a). Dirichlet boundary conditions are then added by fixing 54 nodes in the area separating the two lobes. For the data generation, normal forces of random magnitudes are computed on the liver surface and applied on the hexahedral grid through a mapping in order to generate a data set of $2,000$ samples ($N = 1600$ samples for training and $M = 400$ samples for validation). The metrics obtained on the validation set are reported in Table 7.2. The maximal error is of only $4.07e{-}04\,m$ for a maximal deformation of $0.0536\,m$. The outputs are predicted in only $3\,ms$. In Figure 10.6 are shown some samples of U-Mesh-deformed livers and their corres-



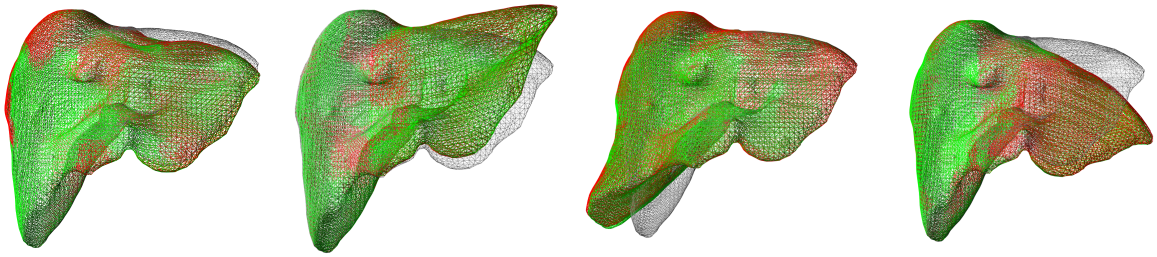
(a) (b)

Figure 7.5: (a) Hexahedral simulation sparse grid of 1109 nodes. (b) Input to the U-Net: regular grid of size $16 \times 15 \times 16$.

c	k	FSS	$\overline{e}$ in m	$\sigma(e)$ in m	pred_t in ms	train_t in min
128	3	1024	2.89e-05	3.41e-05	3	149

Table 7.2: Error measures on a liver of length $0.2\,m$ Immersed in a $16 \times 15 \times 16$ grid. The maximal error is $4.07e-04\,m$ and the maximal deformation of $0.0536\,m$.

ponding relative errors computed at one of the lobe tips. Furthermore, the slope of the regression in Figure 7.8 shows that the increase of the error with the deformation amplitude is also controlled for this scenario.



(a) Relative l^2 norm 3.2% (b) Relative l^2 norm 2.6% (c) Relative l^2 norm 8.5% (d) Relative l^2 norm 2.4%

Figure 7.6: Various liver samples from the testing data set and corresponding relative errors computed on the tip of the deformed lobe. The rest shape of the liver is shown in grey.

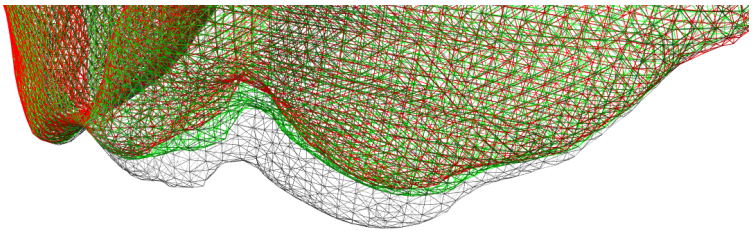


Figure 7.7: Sample with maximal maximal nodal error ($0.0153\,m$) for the hexahedral mesh. The reference solution is shown in red and the U-Mesh predictions is in green. The rest shape is shown in grey.

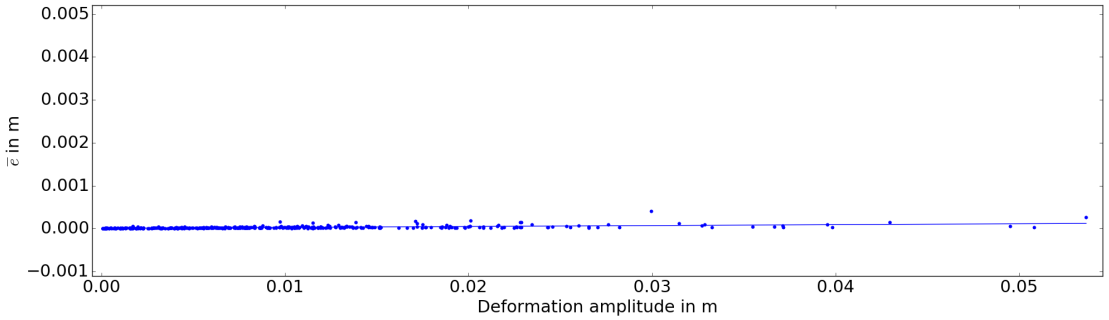


Figure 7.8: Sensitivity of e to the deformation amplitude for the liver. The point cloud represents the error e of all the samples of the testing data set. The regression line of equation $y = 0.0021 \times x$ shows the low sensitivity of the U-Mesh to the deformation range.

7.3.2 Results in augmented surgery

To apply the U-Mesh in order to generate an augmented view of an hepatic surgery, it needs to learn to predict full volumetric displacements from partial surface point clouds that give information about the position of some points of the surface of the liver. These positions can be translated as prescribed constraints. We choose to use a FEM combined with an *Immersed-boundary method* to generate the data sets.

We can assume that during an open liver surgery, half of the surface of the liver is visible to the camera. As depicted in Figure 7.10a, 100 points are uniformly sampled in the visible part of the surface to mimic a point cloud. Then, 100 simultaneous forces of random magnitude and direction are applied to these points in order to generate nearly random displacements. The training data set consists of pairs of $(\mathbf{U_s}, \mathbf{U_v})$ where the input to the network $\mathbf{U_s}$ corresponds to the surface point cloud mapped onto the regular grid. For the same reasons stated in previous section, we limited the size of the data set to $2,000$ samples ($N = 1,600$ for training and $M = 400$ for testing). The reader may refer to Figure 7.9 for examples of the generated

deformations. It is worth mentioning that no patient-specific parameterization of the biomechanical model is required since for homogeneous materials, the relation between the surface and the volumetric displacements is independent of the stiffness of the object [Miller et al., 2013], and only depends on the Poisson's ratio (set to 0.49 as soft tissues can generally be described as incompressible).

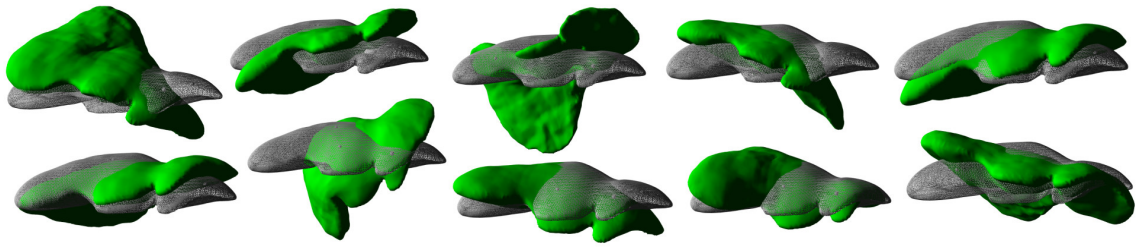
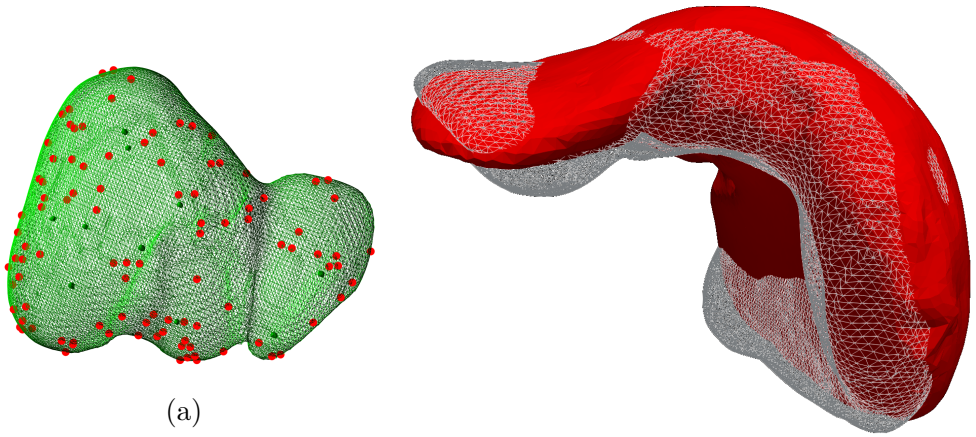


Figure 7.9: Examples of generated deformations to train the network.

***Ex vivo* liver** Once the network is trained, we assess our approach on *ex vivo* human liver data, on which ten markers are embedded to compute target registration errors (TRE). Note that the network is trained on the considered *ex vivo* liver geometry. During the experiments, surface data is obtained with an RGB-D sensor and ground truth data acquired at different stages of deformation using a CT scan. Before computing the displacement field, the point cloud needs to be cropped to the portion of the surgical image that contains the liver. This is done by segmenting the associated color image, similarly as in [Petit et al., 2015]. The RGB-D point clouds can be interpolated onto the regular grid to obtain per-node displacements on the surface and can be given as input to the network that in turn predicts the volumetric displacement fields.

The marker predicted positions are compared to our *ex vivo* ground truth by computing TREs (see Figure 7.10b). The average TRE at the 10 markers is of $7.5\,mm$ with a maximal value of $10.5\,mm$. The solution of the Saint Venant-Kirchhoff model, used to train our model, gives nearly the same error (which was expected) but for a computation time of $1550\,ms$.

***In vivo* liver** With a PyTorch implementation of the U-Net running on a GeForce 1080 Ti, the network can predict the volumetric deformation of the liver in only $3\,ms$. We can then apply this prediction each time the RGB-D camera generates a point cloud. The RGB-D point cloud is then interpolated onto the grid to obtain per-node displacements on the surface (i.e. $\mathbf{U_s}$). Given this input, the network predicts the volumetric deformation, and the next point cloud can be processed. Each new RGB-D point cloud can be fed to the network, thus generating a continuous visualization of the internal structures of the organ. We show some frames of the obtained augmented reality view in Figure 7.11. The quality of the registration seems acceptable in the



(a)

(b)

Figure 7.10: (a)Virtual point cloud on the visible surface of the liver to generate random displacements (top view). (b)The U-Mesh prediction appears in red and the intraoperative CT scan is in gray (side view).

frames on top. However, in the bottom frames, where occlusions appear, the quality is far from being satisfactory. We observe very similar behaviors when using the FEM (both with the Saint-Venant Kirchhoff or the co-rotational model). This means that the issue probably comes from the *iterative closest point* (ICP) algorithm, used for matching the points from the RGB-D point cloud to the surface model. As a matter of fact, our implementation of the ICP does not necessarily handle occlusions and when very large deformations happen between two consecutive frames, the matching fails. Another possible source of error is the model parameterization. Indeed, the liver model is only fixed at nodes that are close to the falciform ligament (whose location is approximate) whereas some other areas of the vascular tree might be stiff enough to consider them as fixed. In order to overcome this source of inaccuracy, in Chapter 9 we propose to estimate such fixations intraoperatively using Kalman filters.

7.4 Conclusion

In this chapter we have shown the ability of the U-Mesh framework in learning complex elastic deformations of any type of geometry. In particular, the U-Mesh learns the mechanical behavior of a liver and generates its deformed state about 500x faster than a reference FE solution. Since the network takes as input a regular sparse grid where displacements are imposed, we have shown an efficient FE *Immersed-boundary method* based on the same hexahedral discretization from which thousands of deformed configurations are generated to train the network. Complex and accurate

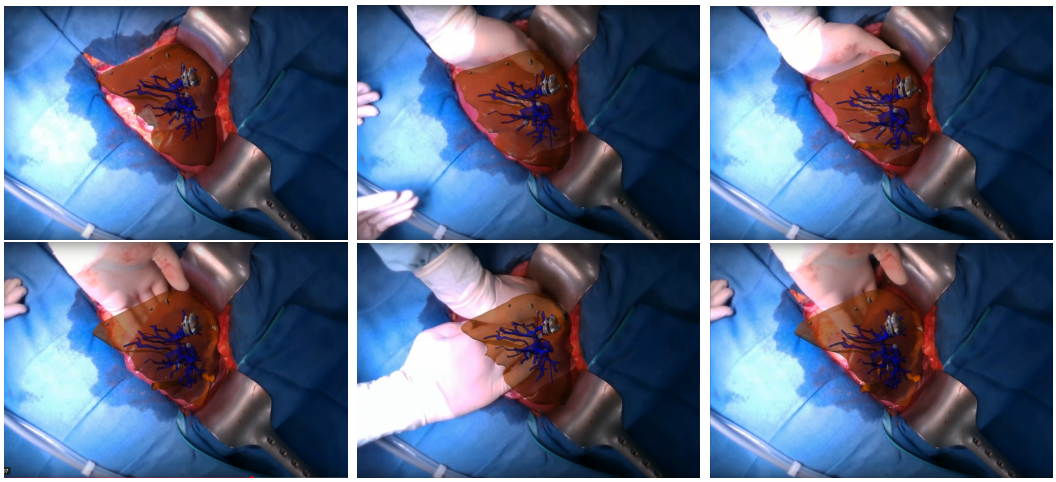


Figure 7.11: Augmented reality frames obtained with U-Mesh predictions. The vascular tree appears in blue, the tumors are in gray and the parenchyma is in brown.

deformation of a preoperative organ model are computed in only a few milliseconds. Driven by surface displacement data, it makes this approach an ideal candidate for providing augmented reality during surgery.

The main limitation of our approach relies in the encoding of the RGB-D point cloud into the U-Net input grid. Each point of the acquired point cloud has to be associated to one (or possibly more) nodes on the hexahedral grid. The problem is that a generic point p_2 from the intraoperative RGB-D point cloud corresponds to a point p_1 on the preoperative liver surface (found using ICP), which is not necessarily a node of the grid (see figures 7.12a and 7.12b). However, our method requires that p_2 coordinates are mapped into grid coordinates. This problem does not have a unique solution. For example, we could set the value of p_2 to the grid node that is closest to p_1 (Figure 7.12a) or spread its value to the 8 nodes of the cell containing p_1 (Figure 7.12b). These two options introduce an approximation error that can be minimized by increasing the grid resolution.

In order to avoid such error, we decided to use the FE shape functions to obtain the nodal grid values. In particular, we computed the FE solution with the Saint-VenantKirchhoff model on the hexahedral grid by imposing the acquired surface displacement. Then, we stored the positions of the grid nodes associated to those cells containing the points of the RGB-D point-cloud to build a virtual point cloud that is in turn fed to the network (see Figure 7.12c). The main limitation of this approach is that we are also giving some volumetric information as input to the network (n_3, n_4, n_6 and n_7 are grid nodes inside the volume), which is not optimal. An alternative solution worth trying is depicted in Figure 7.12d. It consists of

encoding each point of the point cloud as a distance field from each of the grid nodes, similarly as works in [Pfeiffer et al., 2020].

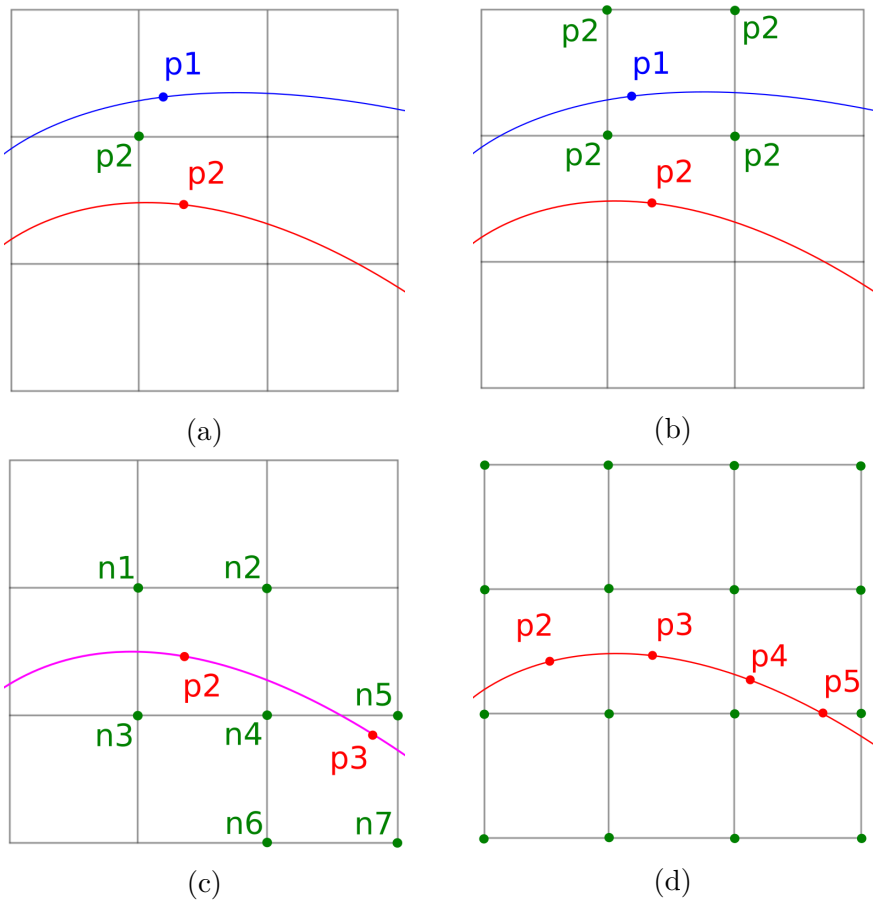


Figure 7.12: Overview of the possible approaches to map a point-to-point displacement in grid coordinates. Blue curve: preoperative surface model. Red curve: intraoperative surface (i.e. the RGB-D point cloud). Purple curve: surface solution computed with the FEM. p_2 is the position of a point in the RGB-D point cloud and p_1 is the position of its corresponding point in the undeformed surface mesh (correspondence found with the ICP algorithm). (a) Set p_2 to the grid node that is the closest to p_1. (b) Set p_2 to the 8 nodes of the cell containing p_1. (c) Store the positions of the deformed grid nodes obtained with FEM for each point of the RGB-D point cloud, and create a virtual point cloud $\{n_i\}_{i=1...n}$ in grid coordinates. (d) For each node in the grid, compute its distance with the closest point in the RGB-D point cloud.

Contributions of this chapter:

- The U-Mesh applied to non regular geometries
- *Immersed-boundary method* to encode the displacement fields in a regular grid structure given as input to the network
- The U-Mesh for augmented hepatic surgery with a simple set-up (one single RGB-D camera)
- Publication: Mendizabal, A., Brunet, J.N., Petit, A., Golse, N., Vibert, E., Cotin, S., 2019. **Physics-based deep neural network for augmented reality during liver surgery.** International Conference on Medical image computing and computer-assisted intervention, pp. 137-145

Chapter 8

The U-Mesh for Ultrasound-guided Breast Biopsy

8.1 Introduction and related work

To demonstrate the generality of our approach, we propose to apply it to a different organ and a different intervention: US-guided breast biopsy. Breast biopsy is the preferred technique to evaluate the malignancy of screening-detected suspicious lesions. To direct the needle towards the target, biopsy procedures are performed under image guidance, normally done with ultrasound (US) probes due to their ability to provide real-time visualization of both the needle and the internal structures [O'Flynn et al., 2010]. However, proper needle placement with US remains a challenging task. First, malignant lesions cannot always be adequately visualized due to the poor image contrast of US. Furthermore, navigation towards complex 3D lesion geometries is commonly achieved using 2D freehand US (FUS) systems, which provide information in a lower-dimensional space [Krücker et al., 2011]. Since highly sensitive preoperative images (such as MRI or CT) can provide accurate positions of the lesions, finding a method to update these positions from real-time US images during an intervention would highly benefit current biopsy procedures. Several commercial and research platforms have implemented image fusion techniques that align preoperative and intraoperative data, exploiting rigid or affine registration methods [Guo et al., 2017]. However, when dealing with breast anatomy, large deformations arise due to compression forces applied by the US probe. To provide accurate probe-tissue coupling and acceptable image quality, an appropriate alignment procedure of the preoperative and US data is required.

Although FE models have been successfully employed for multimodal breast image registration, they have never been applied to registration between preoperative data and intraoperative US, due to difficulties in providing a prediction within real-time constraints [Hipwell et al., 2016]. This is especially true when considering large,

non-linear deformations which involve hyperelastic objects, as it is the case for the breast.

As mentioned in Chapter 4, various techniques have been proposed to simplify the computational complexity of FEM in order to meet real-time compliance. The most optimized approach used to model breast biomechanics is the one proposed by [Han et al., 2013], which relies on GPU-based TLED formulation. Despite the significant simulation speedup achieved, solving the FE system took around $30\,s$, which is still not compatible with real-time. Modelling methods that do not rely on continuum mechanics laws have also been used to approximate soft tissues behavior. Among these, the position-based dynamics (PBD) approach has been used to predict breast lesions displacement due to US probe pressure in real-time, providing comparable accuracy with FE models [Tagliabue et al., 2019]. However, not being based on real mechanical properties, such model requires an initial optimization of simulation parameters to obtain a realistic description of the deformation.

In this chapter we propose to use the U-Mesh framework to estimate breast tissue behavior in US-guided biopsy. Using FE simulations for model training in the context of MRI-US deformable image registration has already been proposed in [Hu et al., 2016], where the authors build a statistical model of prostate motion which can account for different properties and boundary conditions. In the case of the breast, the potentiality of employing machine learning techniques has been already shown in [Martínez et al., 2017], where several tree-based methods have been employed to estimate breast deformation due to compression between biopsy plates. These methods have been trained on 10 different patient geometries with a very specific FE simulation, where the upper plate is displaced vertically towards the lower one.

In this chapter, we propose to train the U-Mesh to predict the deformation of internal breast tissues starting from the acquired surface displacements induced by the US probe. Our network can be seen as a patient-specific model. We train it on a single patient geometry before surgery, with a relatively small amount of training data. However, in contrast to the work of [Martínez et al., 2017], FE simulations that compose the training set are generated with several random input displacements, making our approach able to generalize to different probe positions and compression extents.

Similarly to Chapter 7, we propose to use an *Immersed-boundary method* for generating patient-specific simulations. Results presented in Section 8.3 show the efficiency of the method when applied to both synthetic and phantom data. The contribution of this chapter consists of a novel method to generate a real-time capable soft tissue model to improve target visualization during needle-based procedures. The

position of lesions identified beforehand on preoperative images can be updated from intraoperative ultrasound data and visualized by the surgeon in real-time.

8.2 Method

This chapter consists of a data-driven method to estimate in real-time the displacement of the breast internal structures due to probe pressure during US scanning. In our pipeline, we assume to have a patient-specific geometric model of the breast, obtained from preoperative imaging such as MRI, and to know the position and orientation of the US probe at each time, thanks to a spatial tracking system. If the tracking coordinate system and the coordinate system of preoperative imaging are registered, knowledge about the 3D pose and the geometry of the US probe directly allows to identify the contact surface between the breast and the probe. Since the US probe is represented as a rigid body, we can reasonably assume that when the anatomy is deformed by the probe during the image acquisition process, points on the breast surface below the US probe will be displaced to the same exact extent as the probe itself. As a consequence, our method can predict the displacements of all the points within the anatomy given as input the displacement of the surface nodes in contact with the US probe. The decision of relying on surface displacement inferred from the spatial tracking of the US probe instead of directly tracking surface deformations (through, for example, an RGB-D camera) was taken from the fact that probe-induced deformations are large but local, and the probe itself would occlude most of the deformed surface to the sensor, thus preventing an accurate estimation of the contact surface displacements.

8.2.1 Simulation of breast tissue

The training data set consists of pairs of $(\mathbf{U_s}, \mathbf{U_v})$ where $\mathbf{U_s}$ is the input partial surface displacement and $\mathbf{U_v}$ is the volumetric displacement field. Even though the data generation process takes place in an offline phase, in order to generate enough training data with FE simulations within clinically acceptable times (the intervention can be performed on the day after preoperative scan is acquired), it is important to have simulations that are both accurate and computationally efficient.

We consider the boundary value problem of computing the deformation on a domain Ω under both Dirichlet and Neumann boundary conditions. Let Γ be the boundary of Ω (in our case, Γ corresponds to breast external surface, while Ω represents the entire breast volume). We assume that Dirichlet boundary conditions are applied to Γ_D and are *a priori* known, whereas Neumann boundary conditions are applied to Γ_N, a subset of Γ that represents probe-tissue contact area and changes depending on current US probe position.

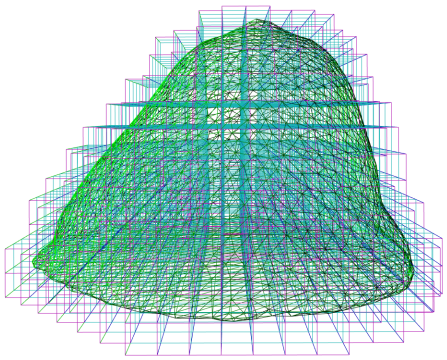


Figure 8.1: Breast surface mesh obtained from a preoperative CT scan Immersed in a hexahedral grid for FEM computations.

We choose to discretize the domain into 8-node hexahedral elements not only for their good convergence properties and lock-free behavior, but also because it is the required structure for the input to the network. To do that, the 3D breast geometry is embedded in a regular grid of hexahedral elements (see Figure 8.1) and we use an *Immersed-boundary method* to correctly approximate the volume of the object in the FE method computations (see Section 7.2 in Chapter 7 for details).

8.2.2 Training data generation

The input to the network corresponds to the displacement $\mathbf{U_s}$ of the points belonging to the breast-probe contact area. The punctual displacements are spread to the nodes of the surrounding cuboid cell through a barycentric mapping and the corresponding volume displacement $\mathbf{U_v}$ is obtained by the previously explained FE approach in response to $\mathbf{U_s}$. The data used to train the network must be representative of the application scenario and must allow the network to extract the pertinent features of the tissue behavior. In order to train our model to estimate breast volume deformation in response to pressure imposed with the US probe, we simulate several random probe-induced deformations using the following strategy:

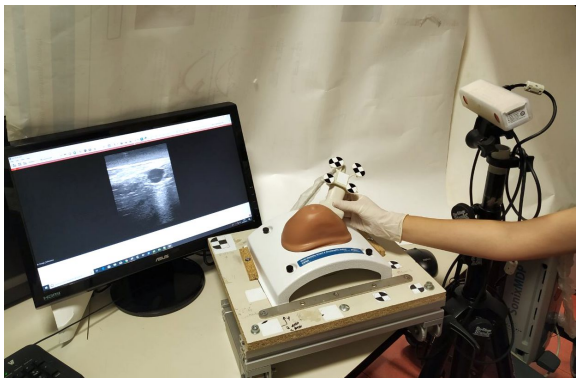
- Select a random node p in the breast surface
- Select an oriented bounding box A centered in point p and normal to the breast surface, whose dimensions match those of the US probe lower surface, which represents current probe-tissue contact area
- Select all the surface points P falling within the box A
- Select as force direction $\vec{d}$ the normal to the surface at point p plus a random angle α ($\alpha \in \left[-\frac{\pi}{4}, \frac{\pi}{4}\right]$)

- Apply the same force $\vec{f}$ of random magnitude ($|f| \in [0.0, 0.8]$) along direction $\vec{d}$ to the P selected points simultaneously
- Store the displacement at the set of points P (input to the network) and the displacement of all the points in the volume (output to the network)
- Repeat the procedure until $N + M$ samples are generated

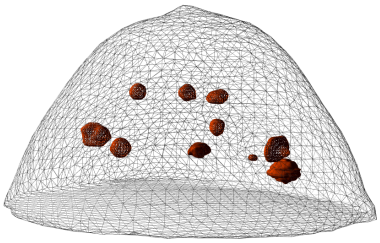
The choice of applying force $\vec{f}$ allowing some angle deviation from normal direction enables us to include in our dataset samples where the probe compression is not precisely normal to the surface, as it can be the case in freehand US acquisitions. The maximal force magnitude (e.g. $0.8\,N$) is set such that the amount of maximal deformation reproduced in the training dataset never exceeds too much that observed in real clinical settings. The described strategy is used to generate the set $\{(\mathbf{U_s}^n, \mathbf{U_v}^n)\}_{n=1}^{N}$ of N samples which is used to train the network, and the set $\{(\mathbf{U_s}^n, \mathbf{U_v}^n)\}_{n=1}^{M}$ of M samples which is left for validation. The training dataset is generated with the SOFA framework [Allard et al., 2007] on a laptop equipped with an Intel i7-8750H processor and 16GB RAM.

8.3 Experiments and Results

The network presented in this work is used to predict US probe-induced deformations of a realistic multi-modality breast phantom (Model 073; CIRS, Norfolk, VA, USA). The 3D geometry model of the phantom surface and 10 inner lesions (diameter of 5-10mm) is obtained by segmenting the corresponding CT image, relying on ITK-SNAP and MeshLab frameworks [Yushkevich et al., 2006, Cignoni et al., 2008]. A Freehand Ultrasound System (FUS) based on a Telemed MicrUs US device (Telemed, Vilnius, Lithuania) equipped with a linear probe (model L12-5N40) is used to acquire US images of the 10 segmented lesions. The dimension of the probe surface is (5x1cm). For each lesion, we acquire US images in correspondence of four different input deformations. The MicronTracker Hx40 (ClaronNav, Toronto, Canada) optical tracking system is used to track US probe in space (Figure 8.2a). The overall probe spatial calibration error is below 1mm (± 0.7147), estimated through the PLUS toolkit [Lasso et al., 2014]. Landmark-based rigid registration is performed to refer the CT-extracted 3D model, the US probe and the US images to the same common coordinate system, exploiting 3D Slicer functionalities [Fedorov et al., 2012]. The registration process does not only enable us to extract the breast-probe contact area, as described in Section 8.2, but also to know in real-time the 3D position of any point belonging to the US image. In this way, it is possible to refer lesions position extracted from US images to the 3D space.



(a)



(b)

Figure 8.2: (a) Experimental setup. From left to right: monitor showing real-time US images; CIRS breast phantom during FUS acquisition; optical tracking system that allows to map the real positions of the CIRS breast phantom and the US probe to the preoperative geometry model. (b) External surface and inner lesions of the CIRS breast phantom.

8.3.1 Predictions on synthetic data

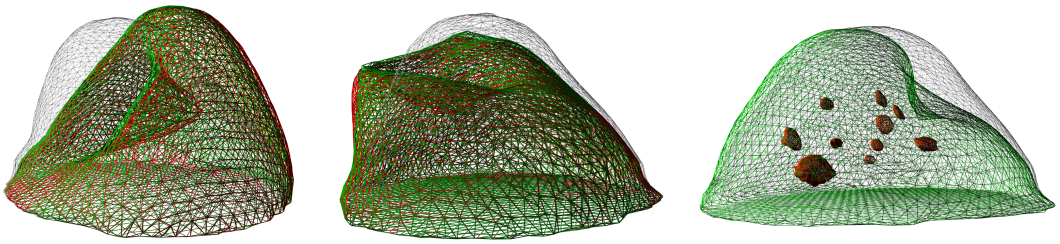
Elastic properties of the physics model used to generate training data are set in accordance with the values estimated in [Visentin et al., 2018] for the same breast phantom considered in this study. However, as we are imposing surface displacements, the values of the elasticity parameters do not affect the displacement field inside the simulated volume as long as the ratio of the different stiffness values is maintained [Miller et al., 2013], thus making the method reliable for any patient specificity. Dirichlet boundary conditions are imposed by constraining the motion of all the nodes belonging to the lowest phantom surface.

Using the method described in Sections 8.2.1 and 8.2.2, we discretized the breast phantom into 2174 hexahedral elements and we simulated several probe-induced displacements. Overall we generated $N = 800$ samples for training and $M = 200$ samples for testing. The U-Net is trained in a GeForce GTX 1080 Ti using a batch size of 4, 100000 iterations and the Adam optimizer. We used a Pytorch implementation of the U-Net. To assess the learning capability of the network, we perform a statistical analysis of the *mean norm error* e over the testing data set following Equation (4.3). We compute the average $\overline{e}$, standard deviation $\sigma(e)$ and maximal value of such norm over the testing data set. The obtained results are shown in Table 8.1. The maximal error is of only $0.266\,mm$ and corresponds to the sample shown in Figure 8.3b. The most striking result is the small computation time required to make the predictions: only $3.14 \pm 0.56\,ms$. In contrast, the FE method takes on average $407.7 \pm 64\,ms$ to produce the solution. Obviously, the resolution of the FE

$\overline{e}$ (mm)	$\sigma(e)$ (mm)	$\max_{m \in M} e$ (mm)	Prediction time (ms)	Total training time (min)
0.052	0.050	0.266	3.14 ± 0.56	278

Table 8.1: Error measures over the testing data set for a breast having 2174 H8 elements, with maximal nodal deformation of 79.09 mm.

mesh could be reduced to accelerate the computations but at the cost of an accuracy loss.



(a) (b) (c)

Figure 8.3: (a) Sample with maximal deformation (79.09 mm). (b) Sample with maximal *mean norm error* (0.266 mm). The green mesh is the U-Net prediction and the red mesh is the FEM solution. The initial rest shape is shown in grey. (c) U-Net prediction on phantom data.

8.3.2 Predictions on phantom data

In our experiments, we consider one lesion at a time and we reposition the US probe on the surface of the breast such that the lesion considered is visible on the US image. In order to validate our model, we manually extract lesions position from US image acquired at rest (i.e., without applying any deformation, when the probe is only slightly touching the surface) and we consider it as a landmark to track. We then impose four deformations of increasing extent for each lesion, and we compare the U-Net-predicted displacement with real displacements extracted from US images. The comparison is performed computing target registration error (TRE) between the predicted position of the lesion and its ground-truth position. The performance of our method is compared to that of the FE model used for data generation. In Table 8.2 are shown the target registration errors for each phantom lesion with respect to the applied deformation. The input deformations are classified into five ranges based on the probe displacements. Displacement ranges indicated as D15, D20 and D25 have a fixed length of 5 mm each and are centered at 15, 20 and 25 mm respectively. D10 and D30 contain the extreme cases under 12.5 mm or above 27.5 mm.

Values in Table 8.2 highlight that the average TRE for all the tumors and for all

Table 8.2: Target registration errors in millimeters for different tumors and different deformation ranges in the breast phantom. The first table is for the proposed method, while the second table reports results obtained with the FE model used for data generation. Not-acquired data is reported as (-).

U-Net predictions

TumorID	D10	D15	D20	D25	D30	Mean	STD
1	-	1.936	2.002	1.506	3.053	2.124	0.569
2	3.211	2.905	4.068	-	4.137	3.580	0.534
3	2.032	-	4.709	7.134	10.90	6.194	3.262
4	0.505	2.225	5.313	5.903	-	3.486	2.217
5	0.932	2.768	3.454	-	4.893	3.012	1.425
6	3.923	6.349	5.625	-	6.724	5.655	1.075
7	3.454	3.864	4.543	6.710	-	4.643	1.255
8	2.422	3.261	4.320	5.136	-	3.785	1.030
9	-	3.928	4.214	4.578	4.858	4.394	0.353
10	5.529	3.272	3.940	4.846	-	4.397	0.860
Mean	2.751	3.390	4.219	5.116	5.761		
STD	1.638	1.294	1.007	1.854	2.788		

FE method

TumorID	D10	D15	D20	D25	D30	Mean	STD
1	-	1.326	2.151	2.075	3.759	2.328	0.887
2	1.956	2.738	3.945	-	4.025	3.166	0.865
3	1.595	-	4.748	7.044	10.932	6.080	3.404
4	0.755	1.991	4.544	5.120	-	3.103	1.795
5	1.029	2.863	3.330	-	4.541	2.941	1.262
6	2.579	3.409	2.871	-	2.337	2.799	0.400
7	2.605	3.219	4.095	6.750	-	4.167	1.582
8	2.695	2.748	4.321	5.411	-	3.794	1.139
9	-	2.745	2.497	2.510	4.193	2.986	0.704
10	2.916	2.542	3.015	3.868	-	3.085	0.485
Mean	2.016	2.620	3.552	4.682	4.964		
STD	0.765	0.593	0.856	1.803	2.757		

the deformations is smaller than $6.194\,mm$ which is comparable to the maximum value obtained with the FE method ($6.080\,mm$). The average error increases with the deformation range just like in the FE method. There is no significant difference between the values of the two tables, meaning that in terms of accuracy, our method is comparable to the data generation method used to train it. In order to compute each deformation, the FE method needs about $407.7\,ms$ whereas the U-Net predicts the deformation in only $3\,ms$.

8.4 Conclusion

In this chapter we have proposed to use the U-Mesh framework to learn the deformable behavior of the breast from numerical simulations based on the *finite-element method*, in order to bypass the high computational cost of the FEM. Our approach represents an interface between precise biomechanical FE modeling (not capable of real-time) and clinical applications requiring both high accuracy and very high speed. We have shown that our framework allows for extremely fast predictions of US probe-induced displacements of the breast during US scanning, achieving comparable accuracy to other existing methods. Therefore, it has the potential to be employed to update in real-time the estimated position of breast lesions identified on a preoperative scan on US images, enabling continuous visualization of the biopsy target, even when sonography fails to render it.

Although the FE model used to train our network does not perform in real-time, its prediction delay of less than $1\,s$ might be considered already acceptable for our specific application. However, such good computational performance is achieved since in this preliminary evaluation we use a very simplistic model, that does not account for heterogeneity or complex boundary conditions happening in clinical cases. Usage of a more complex FE model will certainly cause an increase of computation load. On the contrary, an important feature of our approach is that the prediction time remains close to $3\,ms$ regardless of the grid resolution and of the biomechanical model used for the data generation process. This means that increasing the complexity of the model used to generate the data set will not affect the prediction speed. Moreover, our pipeline allows the method to be insensitive to patient specific elastic properties as it imposes surface displacements. It is worth noting that for inhomogeneous objects, the displacement field still depends on the ratio of the different stiffnesses [Miller et al., 2013]. Another advantage of our method is the easy meshing process. Any geometry can be embedded in a sparse grid and through the use of Immersed boundary simulations the deformations are correctly estimated.

The main limitation of our method remains the training process, which is burdensome and has to be repeated for every new geometry or application. However, we have shown that a limited amount of training data can be sufficient to train a U-Net such that it obtains accurate prediction within clinically acceptable times.

Contributions of this chapter:

- U-Mesh applied to simulate breast deformation
- Data generation strategy adapted to the studied phenomena (simulation of probe-induced displacements)
- Accurate estimation of target location ($\sim 4\,mm$ error) in real-time
- Publication: Mendizabal, A., Tagliabue, E., Brunet, J-N., Dall'Alba,D., Fiorini, P., Cotin, S., 2019. **Physics-based Deep Neural Network for Real-Time Lesion Tracking in Ultrasound-guided Breast Biopsy.** Computational Biomechanics for Medicine Workshop

Chapter 9
Towards Patient-Specific Networks

9.1 Introduction

In Chapter 7, we have shown the potential of the U-Mesh for augmented hepatic surgery. However, we made the assumption that the boundary conditions (BCs), the material properties and even the constitutive law were known *a priori*. As seen in Chapter 5, this assumption can be reductive. The location and the elastic properties of the BCs are patient-specific and are not visible on preoperative images. Therefore, they cannot be identified in advance, which is necessary for the training data generation process. On the other hand, the identification of the elasticity parameters was not crucial in the previously considered scenario as we were dealing with surface displacements as input [Miller et al., 2013]. Nevertheless, if we were to consider a scenario where the input to the network is a traction force, the material parameters do have an impact on the deformation and need to be correctly estimated. Furthermore, the material properties and BCs are relative to a constitutive law that can itself be inappropriate. Therefore, it is important to see how the U-Mesh performs in such situations and how it can be adapted.

This chapter is a proof of concept that explains how the U-Mesh could be integrated in a real clinical routine for patient-specific predictions of the displacement field. It consists of a set of possible research directions with some preliminary results or bibliographic references supporting their feasibility. The variety of parameters intervening in a simulation, may require several strategies in order to adapt the networks to patient-specificity. We will propose here two major solutions relying on two opposite paradigms.

The first one consists of training the network with all possible combinations of pertinent parameters, namely with exponentially large amounts of training data. Such parameters must be sampled and explicitly encoded in the input to the network (in particular the domain of possible geometries, elasticity parameters and even BCs).

The second option, more subtle, relies on *transfer learning* techniques where the net-

work's weights are updated based on patient-specific data acquired intraoperatively. As a matter of fact, when more primary components of the simulation (such as the constitutive law) must be revised, it seems pertinent to explore this alternative as the learned underlying function must be modified. In an ideal scenario, we could use real patient data to retrain our model. An interesting point of this solution is that the parameters do not need to be explicitly identified as they are encoded in the data, meaning that the material properties, the BCs and the constitutive model can simultaneously differ from the ones learned preoperatively. However, as seen previously, collecting volumetric data on a patient without exposing him or her to radiation is in practice very difficult. To overcome this issue, we can generate new synthetic data (where the simulation has been parametrized using intraoperative observations) and retrain the neural network with these new simulations. The BCs and the material properties are relative to a constitutive law (that is difficult to characterize intraoperatively), meaning that their correct estimation can overcome the inaccuracy of the constitutive law (if any). Therefore, we will demonstrate how transfer learning can be used to adapt the network to patient-specific material parameters and BCs. Once the network is trained as described in Chapter 7, we propose to estimate the boundary conditions and the relevant material parameters intraoperatively in order to update the network parameters with patient-specific FE simulations. To do that, we first use an image-driven stochastic assimilation method based on Kalman filtering to identify the BCs on the one hand, and the elasticity parameters on the other hand. Then, the pre-trained model is adapted to the patient specific properties through transfer learning. In this chapter we only show some preliminary results on a cantilever beam to showcase the potential of transfer learning in our applications.

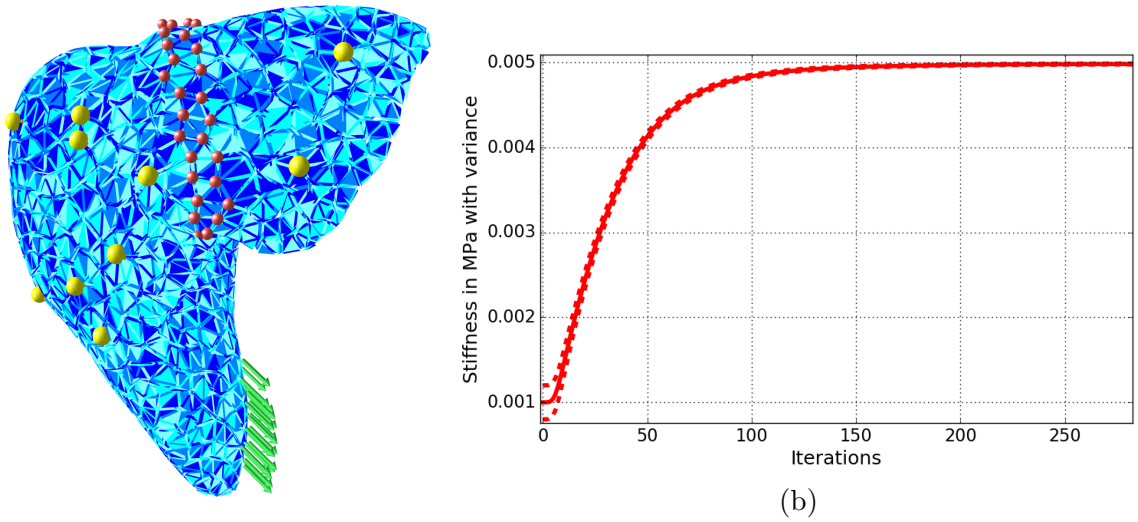
This chapter is divided in three main segments. First of all we will see how to use Kalman filters to identify the BCs on the one hand, and the elasticity parameters on the other hand. Second of all, we will briefly explain how the identified parameters and other patient-specific properties (such as the geometry) can be encoded in the input to a generic network, in order to make patient-specific predictions. Lastly, we will demonstrate how the parameters identified in the first section can be used to update the network's weights using transfer learning.

9.2 Stochastic identification of patient-specific properties

We propose to use the ROUKF presented in Section 5.3, to estimate the value of Young's modulus and the BCs of a liver using observations of the target model. In this section, the ROUKF algorithm is first used to estimate the Young's modulus of

a synthetic liver, and in a second time used to estimate the boundary conditions of an *in vivo* liver.

9.2.1 Elastic modulus



(a) (b)

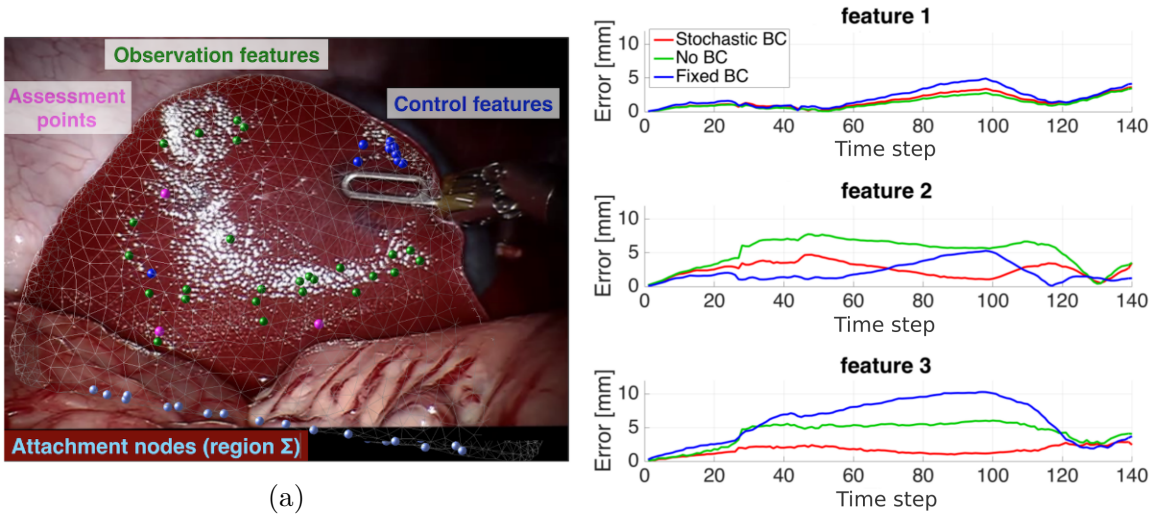
Figure 9.1: (a) Liver simulation mesh made of 11,000 tetrahedral elements. The red points highlight the fixed points (Dirichlet boundary conditions), the yellow points correspond to the observed features, and the green arrows illustrate the direction of the applied forces. (b) Variation of μ and σ for Young's modulus estimation using the ROUKF. The value of the parameter converges to $4992 \pm 15Pa$ in 500 seconds.

In this paragraph, we aim at estimating the value of the Young's modulus of the liver using the ROUKF. This estimation is done using synthetic data, but a similar process can be followed for real data. We build a biomechanical model of the liver, with fixed boundary conditions (red points in Figure 9.1a) to mimic the effect of the falciform ligament and of the vascular tree. A force of fixed magnitude and varying amplitude is continuously applied to one of the liver lobes to generate observations (yellow points in Figure 9.1a). The amplitude of such force follows the sinusoidal function $\frac{1}{2} \times (1 - cos(2 \times \pi \times \tau))$ where τ is a period. In this case, the control features defined in Section 5.3.1 correspond to the force applied (that is known). The Young's modulus is set to $5,000Pa$ in the reference simulation.

For the initialization of the ROUKF, we set μ_0 to $1,000Pa$ and σ_0 to $200Pa$. The state vector contains all the degrees of freedom of the mesh and the parameters to estimate (one parameter in our case). Hence, there are only 2 sigma points which allows a very fast assimilation process to take place as only two evaluations of the model need to be performed at each prediction phase. As depicted in Figure 9.1b,

the value of the Young's modulus reaches rapidly a value close to the ground truth (at iteration 150, $\mu = 4948$ and $\sigma = 97$). The value of the parameter converges to $4992 \pm 15Pa$ in 631 iterations (that is 500 seconds). Such assimilation process could take place before the surgery starts. Note that if the assimilation needs to be done in real-time, the simulations could be parallelized and simplified (we chose here a relatively high mesh resolution).

9.2.2 Boundary conditions



(a)

(b)

Figure 9.2: (a) The first frame of the video sequence with features. (b) Temporal evolution of the assessment error computed for each assessment point.

Apart from material properties, the same approach can be employed to estimate the unknown attachments of the liver. We consider a scenario where an *in vivo* porcine liver is deformed with laparoscopic pincers. A video sequence of 7 s was acquired with a laparoscopic monocular camera inserted in the porcine abdomen inflated with gas. We assume there is a region Σ on which hidden boundary conditions are applied. Before the intervention, a CT scan was collected from the organ's geometry and a FE model was built following the pipeline described in Section 2.2. A tetrahedral mesh having 315 nodes was generated. The obtained model is fixed with elastic springs in a region Σ that is hidden to the laparoscopic camera view. There are 35 nodes in Σ meaning that 35 nodes were attached with springs to mimic the boundary conditions at these specific locations. The elasticity parameters of such springs are modelled as stochastic parameters (see Section 5.3.1). The elasticity values can range from 0 (no attachment) to high values (stiff attachment). A different elasticity is associated with each spring. Known surface displacements are prescribed on a small area of the visible

surface to mimic the effect of the surgical tool based on the control features. The considered scenario falls within the category of displacement-zero traction problems, where the relation between surface and volume displacements is independent of the Young's modulus, for homogeneous materials [Miller et al., 2013]. As a consequence, without lack of generality we set Y to the fixed value of $5,000\,Pa$, which is the average stiffness value for a liver. For the initialization step of the ROUKF, $\mu = 0$ and $\sigma = 0.01$ for each parameter. Three assessment points are placed on the surface of the liver mesh in order to compute the prediction error between the observed data and the model. We ensure that the assessment points are different from the control and the observation features (see Figure 9.2a). In Figure 9.2b is shown the temporal evolution of the prediction error computed over the three assessment points. The error achieved with the stochastic simulation is compared to the error obtained with either fixed BCs (e.g. stiff attachment) or without BCs (e.g. spring stiffness set to zero). Results show that the stochastic simulation leads to smaller errors than the deterministic simulation.

The identified parameters can now be used to make patient-specific predictions by feeding them as input to the network (see Section 9.3). On the other hand, the obtained stochastic simulations can be used to generate new patient-specific data in order to retrain our network. To this end, in Section 9.4 we will explain how we can employ transfer learning methods to make the U-Net able to generalize to new unseen boundary conditions and elastic parameters.

9.3 Patient-specificity as input to the network

In the view of training a general network capable of predicting the displacement field of any liver geometry, with any type of BCs and material parameters, it is essential to teach the network to elucidate such variables. A solution might be to encode all these variables in the input to the network. That way, a universal network can be trained with a large variety of combinations of the sampled parameters. However, exhaustively sampling all the possible combinations would be extremely time consuming and probably useless to the network. Instead, a simple option consists of randomly sampling the parameter's space as done in [Pfeiffer et al., 2020, Pellicer-Valero et al., 2019]. Contrary to the approaches presented in chapters 4, 7 and 8 of this manuscript, the data generation and the training of a generic network would take more than a week as reported by [Pfeiffer et al., 2020]. Nevertheless, such expensive training is only performed once, and the trained model can be used for new liver geometries with new parameters as long as they are identified at prediction time. In this section we will see several strategies in order to encode patient variability in the input to the network.

9.3.1 Geometry and heterogeneity

As seen in Chapter 5, the first step towards patient-specific modeling of an organ is to account for its 3D geometry. The 3D surface of the organ can be segmented on a preoperative CT scan for instance. In order to encode such geometry surface in the input, several solutions have been proposed in the literature.

For instance, [Pellicer-Valero et al., 2019] provided a parameterization of the geometry by registering the liver surface to an average liver model. They performed a non-rigid registration using a modified Coherent Point Drift algorithm in order to express how much each node of the surface must be displaced (in each direction) in order to match the average liver surface. Then, Principal Component Analysis was used in order to reduce the size of the feature vectors to only 27 Principal Components. With this encoding procedure, they trained a feedforward neural network with tens of thousands of simulations on more than 100 liver geometries and obtained good accuracy on unseen liver geometries. Note that this work was only tested with synthetic data.

On the other hand, [Pfeiffer et al., 2020] proposed to express the preoperative volume geometry as a signed distance field. They generated a random 3D surface for each sample (e.g. $80,000$ different geometries) by extruding and remeshing an icosphere. They used a very similar network as ours so the obtained surface meshes must be encoded on a regular grid. To do that, at each grid node, the distance to the nearest surface point was computed. The points inside the organ volume take a minus sign thus leading to a signed distance field encoding the geometry. They trained the network with $460,000$ samples and tested it on real laparoscopic data. The obtained accuracy is not satisfactory yet.

The preliminary solution we tried consisted in encoding the geometry as a binary mask over the input grid (ones on the nodes of the sparse grid and zeros elsewhere). We trained our U-Mesh with ten different geometries generated from a beam to which some cells were removed. We ensured that each geometry reacted in a significantly different manner to a given external force. We generated $160,000$ training samples in total and trained the network for 100 epochs. Each sample $(\{F, M\}; U_v)$ corresponds to an input tensor $\{F, M\}$ of size $n_x \times n_y \times n_z \times 4$ and an output tensor U_v of size $n_x \times n_y \times n_z \times 3$. F is the applied force encoded in the grid (3 values per node), M is the binary mask encoding the preoperative geometry (one value per node) and U_v is the corresponding volumetric displacement. We obtained promising results but we did not extend these results to liver geometries yet.

The overall idea would be to train a general network with various liver geometries encoded using one of the above presented strategies and then, during surgery, make predictions with the patient's anatomy expressed in the input. Moreover,

if we were to use a mask, we could encode material heterogeneity (assuming it is known) by setting the mask value to the normalized Young's moduli. Note that in [Pellicer-Valero et al., 2019], the usage of a binary mask was not recommended as in feedforward neural networks such mask should be flattened into a vector. This would mean that two voxels in the same location from two different livers represent the same feature which is not necessarily true.

9.3.2 Material parameters and boundary conditions

The other patient-specific parameters are the material properties and the BCs. Material properties such as Young's modulus and Poisson's ratio can be given as extra channels in the input. With the U-Mesh, this means giving an input tensor of size $n_x \times n_y \times n_z \times 3 + i$, i being the number of extra parameters. We trained the U-Mesh with variable Young's modulus lying in $[0Pa; 10,000Pa]$. To do that, we sampled the domain and generated $16,000$ samples of random deformations for different values of E. We trained the network for 100 epochs and get small and constant errors when testing on different Young's modulus falling within the training range. An important requirement seems to be the normalization of such values over the sampled domain.

On another hand, fixed boundary conditions could also be given as a binary mask in the input, by setting to one the value of the nodes that are fixed and to zero the free nodes. However, we did not test this idea since having completely fixed nodes is a simplification of our FE simulations. Indeed, a better approximation of the boundaries consists of using springs to model organ's attachments and encoding those in the input seems non trivial. In the following section, we will see an alternative to deal with parameters that are difficult to explicitly encode in the input.

9.4 Transfer learning for patient-specific simulations

Let us assume that we have trained the network for predicting the deformation of a liver following the procedure described in Chapter 7. At this stage, the U-Mesh has learnt the chosen stress-strain relationship with average boundary conditions and material properties. In this context, the geometry is patient-specific as it is obtained from the patient CT scan. Now, using the intraoperatively estimated parameters (see Section 9.2) we can build a patient-specific FE model in order to generate a new small training data set. In this section we will see how to use this data set in order to update the network's weights through transfer learning.

As mentioned in Chapter 4, there exists a correlation between our method and model reduction techniques. There is an important body of work in this area, with a

well-established understanding of the process linking the fast (macro) model to the full (micro) model [De Angelo et al., 2019]. Such theory-driven approaches define how to generate reduced models with adapted parameters that characterize the full (micro-scale) model [Boutin et al., 2017]. Our deep learning approach does something similar by learning the key characteristics (deformation and parameters) of the full-scale model, but using a data-driven approach for this.

When applied in the context of surgery, both approaches share the same limitation. The full model (micro-model) cannot always be correctly parameterized until the surgery has started, as some model parameters are not measurable in preoperative images. In this case, the use of transfer learning methods can offer a natural, data-driven solution for adapting the neural network to a particular patient. For methods based on reduced models, Bayesian approaches are probably a good alternative, as they can estimate material properties from a probability distribution and *a priori* knowledge of the parameter value.

As mentioned in the Section 5.3, boundary conditions have a significant impact on the accuracy of the predictions computed by biomechanical models. However, since they are hard to identify, we want to ensure the robustness of the U-Mesh to the variability of the BCs. We will show that a small amount of data is required to learn patient-specific BCs, when refining a network pre-trained with variable BCs from an appropriate distribution. This could help to significantly reduce the expensive cost of the offline data generation phase. Lastly, since real data can be sparse and noisy, we explore the behaviour of the U-Net when the input tensor $\mathbf{C}$ is highly sparse, and the effect of noise on the quality of the predictions.

9.4.1 Validation metrics

To assess the efficiency of our method, we perform a statistical analysis of the error over the testing data set $\{(\mathbf{C}_m, \mathbf{U}_{\mathbf{v}m})\}_{m=1}^{M}$. Let $\mathbf{U}_{\mathbf{v}m}$ be the ground truth displacement tensor for sample m generated using the FEM described in Section 2.2 and $h(\mathbf{C}_m)$ the U-Mesh prediction. We define the *mean Euclidean error* e between $\mathbf{U}_{\mathbf{v}m}$ and $h(\mathbf{C}_m)$ for sample m as:

$$e(\mathbf{U}_{\mathbf{v}m}, h(\mathbf{C}_m)) = \frac{1}{n}\sum_{i=1}^{n} \left\| \mathbf{U}_{\mathbf{v}m}^{\,i} - h(\mathbf{C}_m)^i \right\|_2 \tag{9.1}$$

where n is the number of nodes of the mesh. We compute the average $\overline{e}$ and standard deviation $\sigma(e)$ of such norm over the testing data set. The *mean Euclidean error* represents the intuitive nodal distance, averaged over all the nodes of the mesh.

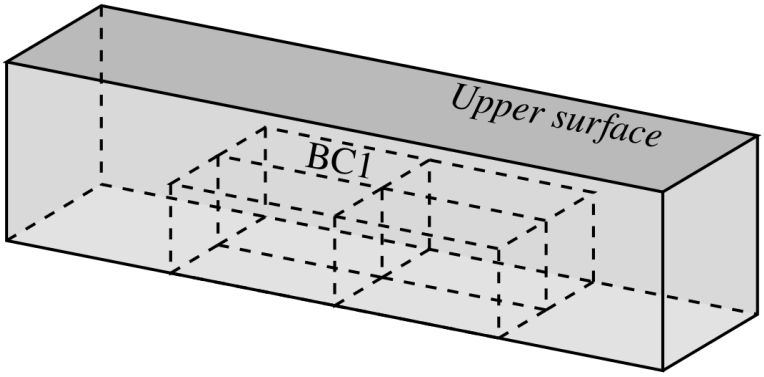


Figure 9.3: Cuboid-like boundary conditions on which the U-Net is pre-trained in strategies 3 and 5. In strategy 6, there are four more cuboids so that the lower part of the beam is fully covered. In strategy 7, the U-Net is pre-trained on BC1.

Strategy ID	1	2	3	4	5	6	7
Training data set #	N_1	N_2		N_3			
Pre training data set	–	–	BC4	–	BC4	BC8	BC1

Table 9.1: Summary of the 7 strategies of interest. "BC4" stands for 4 adjacent cuboids in the middle of the hidden part of the beam. "BC8" stands for 8 adjacent cuboids fully covering the hidden part of the beam (see Figure 9.3).

9.4.2 Beam with hidden boundary conditions

In this section, we compare the accuracy of the U-Mesh either when trained from scratch with up to $16,128$ samples, or when pre-trained on various BCs and refined on the target BCs. We consider a deformable beam (size 4x1x1 m^3, $E = 300$ Pa, $\nu = 0.4$, 500 regular hexahedral elements) subject to fixed boundaries on a rectangular cuboid of its bottom part (see Figure 9.3). The beam follows the Saint-Venant-Kirchhoff behavior described in Section 2.1.3. We generate three different training data sets ($N_1 = 16,128$; $N_2 = 1,209$; $N_3 = 100$) and one testing data set ($M = 4,032$), all drawn from the same distribution. We performed 10 trainings to compare 7 different strategies, summarised in Table 9.1. In strategies 1, 2, and 4, the U-Net is trained from scratch whereas in strategies 3, 5, 6 and 7, the U-Net is refined starting from a network pre-trained with 16,128 data with different boundary conditions (see Table 9.1).

In Table 9.2 are reported the validation metrics computed for each strategy on the same testing dataset (M = 4,032), as well as the index of the best iteration over 200,000 (with a saving step of 5,000). We see that strategy 3 performs better than strategy 2. More impressive yet are the strategies 5, 6 and 7 (especially 5, which, by refining, led to errors comparable to the one obtained with 12x more data without

	e in mm			# training dataset	best iteration
	avg	max	$\sigma(e)$		
1	**0.45**	2.48	0.29	16,128	200
2	**0.71**	2.96	0.47	1,209	180
3	**0.52**	2.66	0.32	1,209	40
4	**3.49**	32.0	3.59	100	200
5	**0.80**	4.85	0.61	100	15
6	**1.11**	5.89	0.80	100	5
7	**0.98**	8.88	0.82	100	5

Table 9.2: Error measures over all seven scenarios. Best iterations are given in thousands. Transfer learning situations are highlighted in red, first and second best results in green and blue.

refining). Furthermore, they are substantially better than strategy 4 where no refinement was done. This is an example of a scenario where the U-Net cannot accurately learn a deformation model from scratch with very few (100) data, whereas it does learn an accurate model in a few thousands iterations using transfer learning in a few minutes (less than 15 minutes for scenario 3 and roughly 5 minutes for scenario 5 to reach best iteration).

The mild differences between the metrics obtained for strategies 5, 6 and 7 show that the data generated for pre-training must be reasonably distributed. Indeed, even though the network benefits from the diversity of BCs encountered in the pre-training stage, it is more efficient when these BCs are close enough to the target boundary conditions. Hence the need for a reasonable distribution.

So far we have seen that refining from an average model significantly reduces the quantity of data required to learn a deformation model. Results in Table 9.3 highlight the fact that it also speeds up the model convergence. Computing more metrics, we found that a good accuracy is reached approximately 20x faster when refining with 100 data than when starting from scratch with either 16,128 or 1,209 samples.

For completeness, we also investigated the case where the constitutive law changed between pre-training and refining stages. For pre-training, we modelled a beam with the linear Hooke's law, and for refining, we chose the Saint-Venant Kirchoff constitutive equation to model the deformations of the beam. In this scenario as well, we observed that transfer learning reduces the amount of data required to reach a given accuracy even when the base equations are complexified.

	e in mm			# training dataset	iteration
	avg	max	σe		
1	**5.74**	27.4	3.73	16,128	5
2	**7.47**	42.1	5.36	1,209	5
3	**0.55**	2.91	0.35	1,209	5
4	**5.25**	44.2	4.71	100	5
5	**0.81**	4.1	0.58	100	5
6	**1.09**	5.37	0.7	100	1.5
7	**0.95**	8.66	0.84	100	1.15

Table 9.3: Error measures at iteration 5000. We relaunched the training of 6 and 7 with a step of 50 iterations to ensure there was no significant overfitting before iteration 5,000, hence the values "1.5" and "1.15" (as a matter of fact, iterations 1,500 and 1,150 were actually slightly better than 5,000). Transfer learning situations are highlighted in red.

9.4.3 New boundary conditions and sparse data

As mentioned in previous paragraphs, real intraoperative data can be sparse and noisy. In this section, we show that the U-Net can still learn models when the training input tensors only contain a sparse view of the displacement $\mathbf{U_s}$ imposed on the upper surface Γ_D. In return, the accuracy is reduced and we show that transfer learning is barely relevant in such an adverse scenario.

We consider the same beam as described in Section 9.4.2, except that the beam is supported on both ends (fixed beam). In order to train the network, we built 2 data sets of sizes $N_1 = 10,080$ and $N_2 = 1,008$ (see Section 9.4.2). Here, every tensor $\mathbf{C}$ contains the values of an imposed surface displacement, on a randomly selected subdomain of the upper surface (in between 13 and 114 non-zero displacements in the testing data set, 67 in average) - see Figure 9.4. We trained the network either directly with N_1 or N_2 samples (strategies 1 and 2), or with N_2 data after a pre-training on a stiffer beam fixed at one end (Young's modulus of 500 Pa). The pre-training was done either with sparse data (strategy 4, same distribution as the refining data set), or dense data (strategy 3, full view of the imposed upper surface displacement $\mathbf{U_s}$).

Figure 9.4: Randomly visible sub-domains of the upper face of the beam (in yellow).

In Table 9.4 are reported the validation metrics at best iteration. The average error with $N1 = 10,080$ is of only $3.03\,mm$ and the maximal error is $57.1\,mm$ - meaning

	e in mm			# training dataset	best iteration
	avg	max	$\sigma(e)$		
1	**3.03**	57.1	3.57	10,080	185
2	**5.69**	101	7.11	1,008	115
3	**7.73**	106	9.02	1,008	145
4	**5.79**	88.3	6.85	1,008	95

Table 9.4: Error measures at best iteration. Transfer learning situations are highlighted in red.

less than 1.5 % of the length of the beam as maximal error. This shows that even though U-Net may learn much more accurate models with dense data, it still deals pretty well with sparse data when provided with a large enough training dataset. We should mention that we obtained very similar results by applying an additive white Gaussian noise of variance $N = 10^{-3}$ m on the testing dataset. With a variance $N = 10^{-2}$ m, the average mean Euclidean error $\overline{e}$ barely exceeds one centimeter. On another note, we see that there is no meaningful difference between the validation metrics of strategy 2 (1,008 data without refining) and strategy 4 (1,008 data with refining). Eventually, except when the data set is very small, we found that refining doesn't enhance accuracy in such a scenario. What is more, these results highlight the importance of pre-training the network with sparse data whenever the refining data is sparse. We further investigated the case where only very few data ($N_3 = 100$) are available, and found that it was not sufficient (with or without pre-training), although the refined model was more accurate. What remains valid is that in any scenario, the U-Mesh maintained a better accuracy with transfer learning in the first thousands of iterations. Reiterating these tests with sparse data without modifying the Young's modulus between pre-training and training stage corroborated these results.

9.5 Conclusion

In this chapter we have proposed several strategies to fulfill the real-time and precision requirements of patient-specific augmented reality. Based on *a priori* knowledge of the biomechanics of the organ, we select a constitutive model describing the relation between stresses and strains. Such relation is heavily affected by patient-specific properties such as boundary conditions and material characteristics. While obtaining these properties preoperatively may be troublesome, having information about them intraoperatively can be straightforward.

In our approach, the parameters of the preoperative *finite-element* or deep learning

models are updated based on intraoperative observations exploiting Bayesian filtering. The obtained parameters can be directly given as input to the network or can be used to retrain it with new data-driven simulations. In the first scenario, expensive data generation and training phases must take place in order to allow the network to generalize to any new liver. This brute force solution has led to interesting results [Pfeiffer et al., 2020, Pellicer-Valero et al., 2019] so it seems worth investigating. On the contrary, employing transfer learning to update a deep learning model is a more subtle solution, that would probably require further research efforts. When using transfer learning, not only the elasticity parameters and the boundary conditions can be changed simultaneously, but also the constitutive model itself. An interesting point of using deep neural networks is that the parameters do not need to be explicitly identified as they are encoded in the data (unless required as input as in Section 9.3). Hence, the network builds its own representation and through transfer learning, the weights of the network can be modified to match the targeted function. Moreover, we have reasons to believe that the U-Net learns local correlations in the displacement field rather than an overall model only. As a consequence, if the pre-trained model represents an average liver, transfer learning should not break the constitutive model learned previously. Note that the variability between livers can be high but it will always vary in a bounded range. For this reason, we believe that transfer learning is the key to an accurate and fast simulation of the deformations of a liver.

Note that when using Bayesian filtering, each parameter can only be modified individually. Indeed, in our pipeline using Kalman filters, in order to estimate the elasticity of the boundary conditions, the Young's modulus of the material needs to be fixed (and *vice versa*). A simultaneous estimation of both sets of parameters would be more complicated (yet possible), less precise (variance of the stochastic parameters will remain high) and would require very tedious fine-tuning of the filter. An alternative would be to use deep neural networks to estimate simultaneously the boundary conditions and the material parameters, but this might require a sequence of deformations as input.

Contributions of this chapter:

- Intraoperative identification of the organ's boundary conditions and stiffness for patient-specific *finite-element* simulations
- Proof of concept: transfer learning to adapt an average network to patient specificity at the beginning of the surgery
- Publication: Mendizabal, A., Tagliabue, E., Hoellinger, T., Brunet, J.N., Nikolaev, S., Cotin, S. (2020) **Data-driven simulation for augmented surgery.** Advanced Structured Materials, vol 132, Springer

Chapter 10
General conclusion

The objective of this thesis work was to come up with new strategies for computing real-time and accurate deformations of a soft structure. Without lack of generality, we considered the deformation of soft tissue in medical applications. In virtual laparoscopic training, surgeons practice on a completely virtual environment thanks to real-time simulations accounting for the interactions between surgical tools and the tissues. As an additional example, in the context of augmented surgery, in order to accurately locate the internal structures of an organ, a 3D model of the anatomy is overlaid on the surgical view by performing an elastic registration between preoperative and intraoperative images. Also in image-guided biopsy or robotized procedures, having a better control on the interactions of the robot with soft tissues can improve the outcome of the procedure and the safety to the patient.

Throughout this manuscript, we showed how combining finite-element simulations with machine learning techniques can help in robotized interventions and surgical guidance. For instance, in the context of robotic assisted interventions, where the knowledge of the force applied by the robot is a key feature for the safety of the patient, we proposed an image-based method for estimating such force without the need of any force sensor. We used a simple fully connected neural network for force classification, where the input to the network is an intraoperative image of the deformed tissue. The network was trained with synthetic data generated with a patient-specific *finite-element* model of the considered organ.

The *corpus* of this work relies on the U-Mesh framework: a method fulfilling the real-time and precision requirements of patient-specific augmented reality. Based on a U-Net architecture, it can learn the relation between a constraint and a deformation for various geometries and make predictions with high accuracy in a very short amount of time regardless of the size of the problem. To make predictions on irregular geometries, we proposed an *Immersed-boundary method* to encode the displacement fields in regular grids handled by the network. In particular, we showcased the potentiality of our method in open liver augmented surgery in humans, with very little equipment (one single RGB-D camera). Based on a preoperative

3D model of the organ, we used the U-Mesh as a non-rigid registration method in order to fit the intraoperative deformations. Our deep learning model is driven by a surface point cloud acquired with one single RGB-D camera, thus allowing for a markerless and radiation-free setup for building up an augmented reality. Our approach could be integrated in any operating room and has great potential in complex liver surgeries where small tumors must be resected. To demonstrate the generality of our pipeline, we also applied the U-Mesh in the context of US-guided breast biopsy. The objective was to track the internal tumors in real-time accounting for the large deformations induced by the US probe. To train our network, we simulated probe-induced compression using the *Immersed-boundary method*. The reached target registration errors ($4\,mm$ in average) was comparable to that obtained with *finite-element* numerical simulations not capable of real-time.

In all the presented scenarios, the U-Net is trained with synthetic data based on *finite-element* simulations. While some aspects of the simulation can be set preoperatively (such as the organ's geometry and average location of ligaments for instance), some others (such as material elasticity and boundary conditions) can only be identified intraoperatively and may require further effort depending on the demanded accuracy. To this end we proposed to estimate such patient-specific parameters based on intraoperative observations through Bayesian filtering. The obtained patient-specific simulations can be used to update our deep learning models employing *transfer learning*. In a similar direction, [Chakraborty, 2020] proposes to train a low-fidelity physics informed model, and refine it with few high-fidelity data exploiting *transfer learning*.

Despite the promising results of our method, there are some limitations worth mentioning. The accuracy of the U-Mesh is only guaranteed as far as the inputs are included in the range of the training data sets. In the same manner, we are restricted to the geometry used to train the network (unless encoded in the input). This is due to the fact that neural networks are not good at extrapolating, hence the importance of a good sampling of the input domain when generating the data sets.

Moreover, even if the choice of our network was motivated by its similarities with model order reduction techniques, we cannot claim that it was the best and only choice. Indeed, we have tried a simple fully-connected network and it seems to produce similar results in terms of accuracy while being about three times faster to train. Nevertheless, we observed that the U-Net has better extrapolation capacity (thus better abstraction of the problem) than the fully-connected networks. This means that even if it is not possible to make very accurate predictions when applying a force somewhere out of the sampled input domain of the training data set, the U-Mesh performs "less bad" than other networks.

Another limitation of the method is the expensive offline phase. The data generation can be extremely time consuming in particular when considering large meshes or when more complex input sequences are needed. It goes without saying that the larger the data sets, the longer the training. Hence it is important to build smart data generation strategies to cover all the force ranges without being exhaustive (to reduce data generation and training times). A possible alternative would be to include in the dataset only those deformation modes which represent significantly different deformations. Moreover, we performed a preliminary study on the sensitivity of the method to the Young's modulus variations in the training data set. We noticed that U-Mesh is more accurate when dealing with stiffer objects. This is an important result to keep in mind for an eventual *smart data generation.* Indeed, in a low deformation regime fewer data are needed to reach good accuracy. In a similar direction, we noticed that when using *transfer learning*, pretraining the network on a slightly softer object than the targeted average, accelerates the convergence.

Future research directions In this thesis work, we have shown the potentiality of using physics-based deep neural networks in *computer-assisted interventions*. We have proved that the accuracy of the predictions satisfies the precision requirements of medical applications such as targeting small tumors. The large field of possibilities and the trendiness of our topic open space for new exciting research directions.

In particular, if we aim at integrating the U-Mesh in any operating room for augmented surgeries, it seems worth investigating how to generalize our network to any kind of geometry using more advanced strategies than those proposed in Section 9.3. Indeed, encoding the geometry seems to be a delicate yet important point, in particular when using regular grids. It is non trivial to express a discrete surface in a grid without introducing approximation errors. In the view of an optimal data generation process, we would like to explore statistical atlases to sample the geometry space in a meaningful manner as well as the organ attachments as done in [Plantefève et al., 2014].

Even if the obtained target registration errors seem compatible with oncological surgery needs, we could further improve the *finite-element* modeling used to generate our training data set. Since the data generation takes place in an offline phase, we could enrich our models as much as needed. For example we could include incompressibility, account for heterogeneity, use quadratic hexahedral elements or simply refine the mesh. If, besides the displacement field, we also want to estimate the stress distribution within the organ, we could use more complex hyperelastic laws or expand our model to account for viscoelasticity. Handling models such as Ogden or Mooney-Rivlin would require no change to our method. Theoretically the architecture of the neural network is independent of the mechanical model but in

practice, if the complexity of the model increases it might be relevant to deepen the network or enlarge the training data sets. On the other hand if we were to handle viscoelasticity, we would need to include a time term and the current state of the system in the network input as done by [Meister et al., 2018].

In order to improve the robustness and accuracy of the method, we also plan to revise the surface point matching between the preoperative and the intraoperative surfaces. For now, we are using the *iterative closest point* algorithm but it might be relevant to also let the network guess the surface correspondences as done in [Pfeiffer et al., 2020]. However, an accurate initial rigid alignment of the surfaces seems crucial for such method to work, as the network produces a result in one iteration. Conversely, we could use the U-Mesh in an iterative process where at each iteration, a displacement can be imposed on a grid node based on the position of its closest point in the RGB-D point cloud. In a similar direction, we plan to integrate multiple RGB-D cameras in our set up, in order to improve the quality of the point cloud describing the surface deformation (larger field of view and less noise), or to use the second camera point cloud for validation purposes.

Another scenario worth investigating is the case of contacts between anatomical structures. Interaction between objects can be seen as external forces applied to their surfaces (the alternative option being to solve interactions through position constraints). Assuming we have two objects embedded in two U-Mesh grids, we can compute their motion until a contact is detected and then apply a simple penalty-based contact response. This contact response is a force applied on the surface of each object to cancel out their interpenetration. Using this force, we could then compute the deformation of each object following our method.

The work presented throughout this manuscript is an interface between precise biomechanical modeling (not capable of real time) and clinical applications requiring both high accuracy and very high speed. Even if there is still room for improvement, our pioneer method has proved successful in scenarios such as augmented surgery and image-guided biopsy.

Part III

Brief summary in french

Titre: Combinaison entre simulation numérique et apprentissage automatique – Applications à la formation, la planification préopératoire et l'assistance peropératoire

10.1 Introduction

Dans un contexte de chirurgie augmentée, il est nécessaire de réaliser un recalage élastique des données préopératoires sur la vue intraopératoire de l'organe. L'objectif est de superposer en temps réel un modèle virtuel à la vue du champ opératoire, afin de visualiser les structures internes de l'organe (tumeurs, réseau vasculaire etc) qui ne sont pas naturellement visibles pour le chirurgien. Le geste chirurgical est ainsi mieux maîtrisé et l'intervention peut être planifiée de façon plus adaptée. Faire ce recalage dans le cas des tissus humains et avec des données intraopératoires très clair-semées, relève plusieurs défis. Par exemple, dans un contexte de chirurgie hépatique, seulement 30% de la surface du foie est visible à cause de la taille de l'incision ou le champ réduit de la caméra laparoscopique [Plantefeve et al., 2016].

Pour modéliser la déformation des tissus mous, la méthode des *éléments finis* (EF) est la technique préférée de part sa capacité à résoudre des problèmes complexes, sa fiabilité et sa robustesse. Pour répondre aux critères de précision de la chirurgie guidée par l'image (erreur de recalage de l'ordre de 5 mm [Ruiter et al., 2006]), le model doit prendre en compte les spécificités de chaque patient notamment d'un point de vue anatomique mais aussi biomécanique et physiologique. Par exemple, les fixations de l'organe ainsi que ses propriétés biomécaniques jouent un rôle très important dans le calcul de sa déformation. Par ailleurs, pour qu'il soit pertinant, le modèle doit faire des prédictions en temps-réel, ce qui est difficile de garantir en pratique surtout lorsque des déformations non-linéaires sont impliquées.

De nombreux travaux ont visé à repousser les limites de temps de calcul des méthodes EF. Très souvent, il s'agit d'un compromis entre temps de calcul et précision. Lorsqu'on cherche à modéliser des petits déplacements, les hypothèses de linéarité de la loi de comportement accélèrent considérablement les calculs. Par contre, lorsque les déformations sont plus importantes, ces simplifications ne sont plus représentatives de la réalité. Une solution serait alors d'utiliser les EF co-rotationnels qui prennent en compte les non-linéarités géomètriques en temps-réel [Petit et al., 2018, Haouchine et al., 2013a]. Cependant, cette approche ne s'étend pas à des lois de comportement plus sophistiquées comme les lois hyperlastiques pour lesquelles, d'autres alternatives ont été proposées. Par exemple, dans [Marchessau et al., 2010], les auteurs utilisent la méthode dite MJED (Multiplicative Jacobian Energy Decomposition) afin de produire en temps-réel les deformations d'un foie hyperelastique, poreux et visqueux. Aussi, les auteurs de [Miller et al., 2007] ont

introduit les TLED (Total Lagrangian Explicit Dynamics) garantissant le temps-réel lorsqu'il sont associés à des schémas d'intégration explicite et des solveurs GPU [Joldes et al., 2010].

Plus récemment, l'emmergence des méthodes d'*apprentissage automatique* (*machine learning* (ML) en anglais) ont permis d'approximer une fonction quelconque, du moment qu'il y a suffisament de données d'entrainement, et ce, sans aucune connaissance a priori du problème. Aussi, ces méthodes font des prédictions en des temps très courts ce qui les rend très interessantes pour nos domaines d'application. En effet, le comportement biomécanique d'un tissus mous peut être implicitement encodé dans un model ML, pour prédire en temps-réel les déformations d'un organe en réponse à une traction de surface [Morooka et al., 2008, Tonutti et al., 2017, Rechowicz et al., 2013, Mendizabal et al., 2019] ou bien à un déplacement imposé [Pfeiffer et al., 2019, Brunet et al., 2019, Lorente et al., 2017, Mendizabal et al., 2019]. Cependant, la précision de la prédiction d'un réseau est fortement liée à la qualité et à la quantité des données d'entrainement. Dans l'idéal, il faudrait une infinité de données réelles et non bruitées du phénomène à approximer, ce qui est en pratique très difficile à assurer. C'est pour cela que les simulations EF peuvent être exploitées pour générer des bases d'entrainement représentatives de la réalité.

Parmis les méthodes de ML, l'utilisation des réseaux de neurones (RN) a considérablement augmenté. Par exemple, Tonutti et al. utilisent un RN pour prédire les déplacements de tumeurs cérébrales en réponse à des forces. De même dans [Rechowicz et al., 2013] la déformation de la cage thoracique est prédite par un RN. Dans ces deux papiers, seul le déplacement surfacique est estimé. Morooka et al. propose les RN pour estimer la déformation volumique d'un foie. Il utilise une ACP (analyse en composantes principales) pour compresser les modes de déformation des sorties du réseau dans le but d'accélérer l'entrainement. Leur model produit des résultats satisfaisants mais a uniquement était testé sur des données simulées.

Afin de répondre aux contraintes de précision et de temps-réel, on suggère de combiner des simulations EF avec des RN pour modéliser le comportement biomécanique des tissus humains. Cette idée s'intègre très bien dans un contexte de chirurgie guidée par l'image, puisqu'il est ainsi possible de collecter beaucoup d'information lors d'une intervention, et ainsi d'apprendre continuellement à améliorer la modélisation. En particulier, nous proposons la plateforme U-Mesh qui permet de prédire en temps-réel les déformations d'un organe hautement déformable comme le foie avec une précision permettant de guider les chirurgiens au cours des interventions où il est essentiel de suivre cette déformation (par exemple lors d'une biopsie). Les réseaux sont entrainés avec une très grande quantité de données générées avec la méthode

des EF. Pour garantir des modèles patient-spécifiques, on combine simulation et inférence bayésienne afin d'identifier les paramètres des modèles EF.

Le manuscript de thèse est divisé en 7 chapitres principaux. Les deux premiers pausent les bases théoriques des sujets que l'on considère, à savoir la formulation du problème bioméchanique et sa résolution par la méthode des EF, ainsi que les fondements théoriques des réseaux de neurones qui seront utilisés. Le chapitre 4 constitue un projet de classification de force lors d'interventions robotizées. Le chapitre 5 présente la plateforme U-Mesh qui est le coeur de cette thèse. Les chapitres 6 à 8 comportent les applications de U-Mesh dans des contextes chirurgicaux.

10.2 Estimation de force avec des réseaux de neurones

Les maladies ophtalmologiques nécessitant des injections intravitréales sont en hausse avec plus de 4 million d'injections dans le monde en 2014 [Ullrich et al., 2016]. C'est pourquoi plusieurs travaux de recherche proposent de robotiser la procédure [Ullrich et al., 2016] afin de libérer les cliniciens et permettre aux hopitaux d'assurer une demande toujours en hausse. Pour designer un tel système, il est nécessaire de développer un robot de précision pour assurer l'efficacité de l'injection ainsi que la sureté du patient. En effet, l'injection doit être faite dans une région très petite se situant entre la cornée et la sclère, sinon l'oeil risque d'être endommagé. Aussi il est important de connaître la force appliquée par l'aiguille sur la sclère afin de garantir la sécurité du patient [Jagtap et al., 2004]. Les senseurs de forces étant couteux et compliqués à utiliser en pratique [Haouchine et al., 2018, Haidegger et al., 2017], il est préférable d'estimer les forces en se basant sur des images intraopératoires [Mura et al., 2016, Haouchine et al., 2018].

Dans ce chapitre on propose d'utiliser un réseau de neurones pleinement connectées (c'est à dire sans couche de convolution) pour classifier la force exercée par l'aiguille en se basant uniquement sur des images OCT de la sclère déformée. En effet pendant l'intervention, il est possible d'acquérir des images OCT 2D de la sclère (voir les images en noir et blanc de la figure 10.2) en capturant la déformation induite par l'aiguille. L'objectif étant d'identifier le moment où le robot excerce une force trop importante, on se propose de classifier les forces en trois rangs :

- rang 0 : peu ou pas de déformation,
- rang 1 : la sclère se déforme mais il n'y a pas de danger pour le moment,
- rang 2 : la sclère se déforme considérablement et on est sur le point de percer la sclère. Il est préférable de rétrograder l'aiguille.

Comme citer précédemment, la qualité de la prédiction d'un réseau de neurones dépend fortement des données d'entrainement. On propose donc de construire un modèle bioméchanique de la sclère afin de générer autant de données que nécessaire. Pour cela on modélise la sclère comme une demie-sphère avec une épaisseur non nulle, fixée aux extrémités (conditions aux limites de Dirichlet) et on applique une pression intraoculaire (IOP) (voire figure 10.1). Pour simuler la force exercée par l'aiguille, on applique une force locale en une zone petite autour de l'extrémité de l'aiguille. On résout les équations obtenues avec la méthodes des EF. Pour cela, on discrétize la demie-sphère avec des élèments hexahédriques (H8) et on cherche l'équilibre des forces (internes et externes) avec un algorithme de Newton-Raphson.

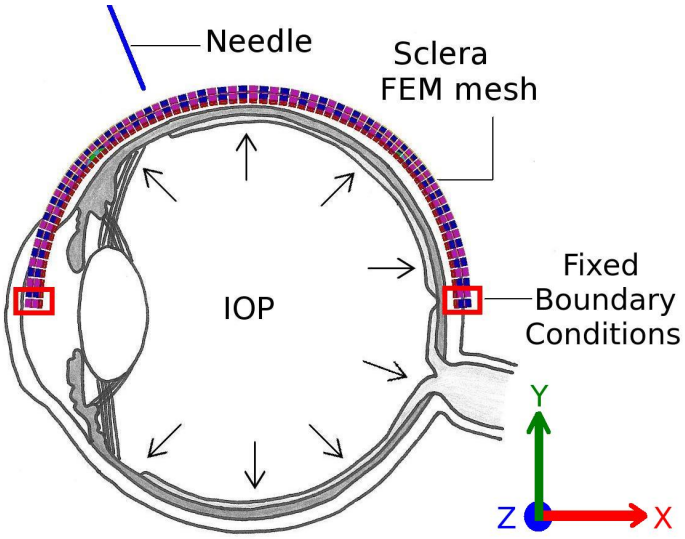


Figure 10.1: Modéle simplifié de la sclère.

La paramétrization du modèle méchanique ainsi que la géométrie du modèle sont très importantes pour que les déformations soient représentatives de la réalité. Pour commencer, l'épaisseur de la sclère joue un rôle non négligeable : pour une force donnée, une sclère épaisse se déformera moins qu'une sclère fine. On va donc faire varier l'épaisseur de la sclère de 400 à 800 micromètres [Olsen et al., 2002]. D'un autre côté, la pression intraoculaire influence aussi la déformation. Elle peut être mesurée avec un tonomètre disponible en consultation ophtalmologique. Dans nos expériences, on utilise des yeux de cochons postmortem qui ont des pressions intraoculaires négligeables (2 mmHg). On peut donc la considérer comme constante dans nos simulations. Un autre paramètre important est la raideur de la sclère (le module d'Young) qui est variable selon les espèces de cochons et selon les conditions expérimentales. On utilise donc des filtres de Kalman pour estimer sa valeur afin de paramétrer nos simulations.

Notre base de données d'apprentissage consiste en des couples (F, I) avec F, le rang

de force appliquée et I, l'image de la déformation résultante. L'image est obtenue en prenant une section 2D du modèle EF 3D. La figure 10.2 montre des exemples d'images OCT réelles vs images simulées pour les trois rangs de force. Une fois les données générées, on peut entrainer le RN. Il s'agit d'un réseau de classification avec deux couches cachées de 600 neurones chacune et une couche de sortie avec 3 neurones (un neurone par classe ou rang de force). L'entrée du réseau correspond à l'image OCT (réelle ou simulée) et la sortie (ou label) correspond au rang de force appliquée.

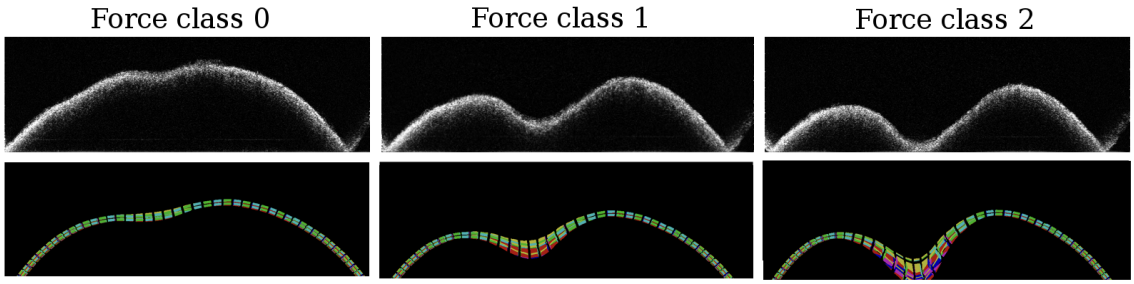


Figure 10.2: Images réelles OCT de la sclère ainsi que les simulations correspondantes pour les trois rangs de force.

Nous avons testé notre méthode sur des yeux de cochons avec 93% de précision dans les prédictions. Néanmoins, pour inclure cette méthode dans une procédure clinique, il faudrait impérativement inclure une valeur de pression intra-oculaire spécifique au patient, ce qui rendrait le modèle biomécanique plus compliqué. De plus, la mesure de cette pression est souvent biaisée par la raideur de l'oeil. Il faudrait donc identifier ces deux paramètres conjointement lors de l'assimilation avec les filtres de Kalman.

10.3 Plateforme U-Mesh: estimation du champ de déplacement avec des réseaux de neurones

Il existe de nombreuses applications où il est nécessaire de calculer les déformations de structures non-linéaires en temps-réel. Comme vu en introduction, la méthode des EF est la plus utilisée pour résoudre ce type de problèmes. Cependant, dès lors que l'on considère des matériaux non-linéaires, il s'avère difficile d'assurer la contrainte de temps réel. Pour accélérer les temps de calculs, plusieurs méthodes ont étaient proposées comme la *décomposition de domaine* [Haferssas et al., 2017], le calcul parallèle sur GPU [Allard et al., 2007, Johnsen et al., 2015] et la *réduction de modèle* [Niroomandi et al., 2008, Ryckelynck, 2005, Goury and Duriez, 2018]. Plus récemment, les méthodes d'*apprentissage automatique* ont été proposée pour modéliser la méchanique des objets [Lorente et al., 2017, Tonutti et al., 2017, Morooka et al., 2008] pour de petites déformations et un ensemble de forces restreint.

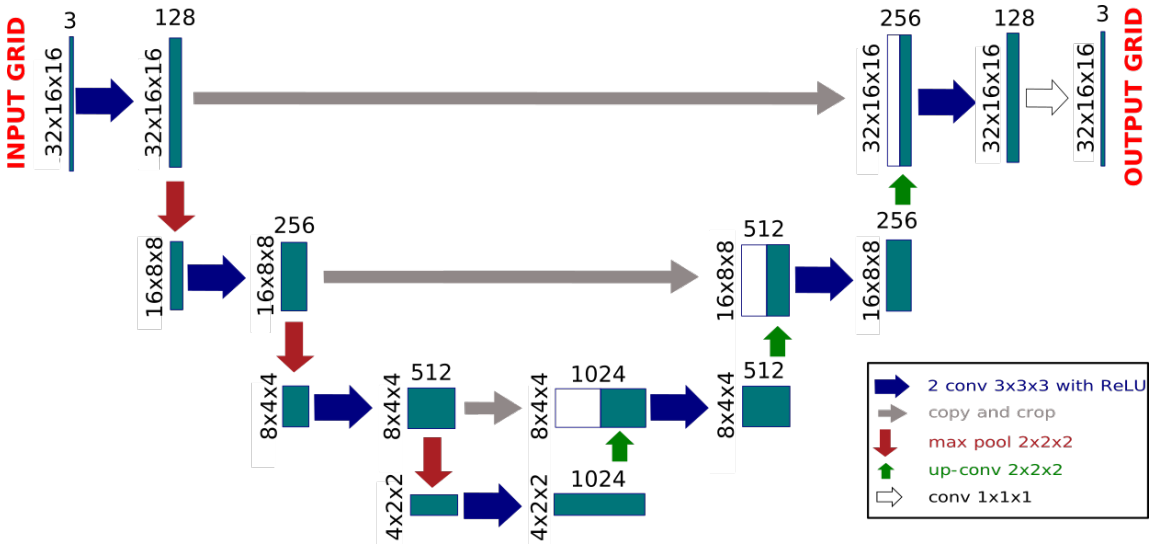


Figure 10.3: Architecture de réseau de la plateforme U-Mesh.

Dans ce chapitre on présente la plateforme U-Mesh qui produit la déformation de géométries simples pour de grandes déformations avec un nombre variable de forces en entrée. On a choisie une architecture de type U-Net [Ronneberger et al., 2015], pour ses similarités avec la réduction de modèle (voire figure 10.3). Il s'agit d'un réseau de type encodeur-décodeur avec des filtres de convolutions pour extraire les traits d'intérêts des entrées du réseau.

Pour entrainer un tel réseau, on génère une base d'apprentissage avec les EF, constituée de couples (F, U) où F est l'entrée du réseau et correspond à la force appliquée, et U est le déplacement volumique résultant (sortie du réseau). Pour cela, on construit un modèle EF de l'objet traité et on applique des forces aléatoires à la surface de l'objet. À chaque simulation, on enregistre le couple (F, U). Un détail important pour que la méthode fonctionne, réside en la nécessité d'encoder les forces et les déplacements dans des grilles régulières. En effet, le réseau U-Net nécessite une structure de grille pour que les filtres de convolutions puissent extraire des informations spatiales cohérentes. On utilise donc des grilles héxahédriques pour nos simulations EF. Ainsi, les nœuds de la grille contiennent le déplacement ou la force des nœuds du maillage EF.

On a testé la méthode sur des géométries de références en méchanique, satisfaisant la structure de grille régulière. Tout d'abord nous avons modéliser le comportement d'une poutre à section carrée, fixée en une de ses extrémités. Une ou plusieurs forces sont appliquées aux nœuds de la face supérieure et données au réseau en entrée. Nous avons aussi modéliser le comportement d'un L. Sur la figure 10.4, apparaissent en vert les solutions EF et en bleu les prédictions U-Mesh. La différence est quasi imperceptible à l'oeil nu. En effet, l'échantillon avec l'erreur maximale (image de

gauche) correspond à une norme *l2* relative de 5.1% à l'extrémité libre de la poutre. Le résultat le plus marquant est que les prédictions sont faites en 3 ms seulement.

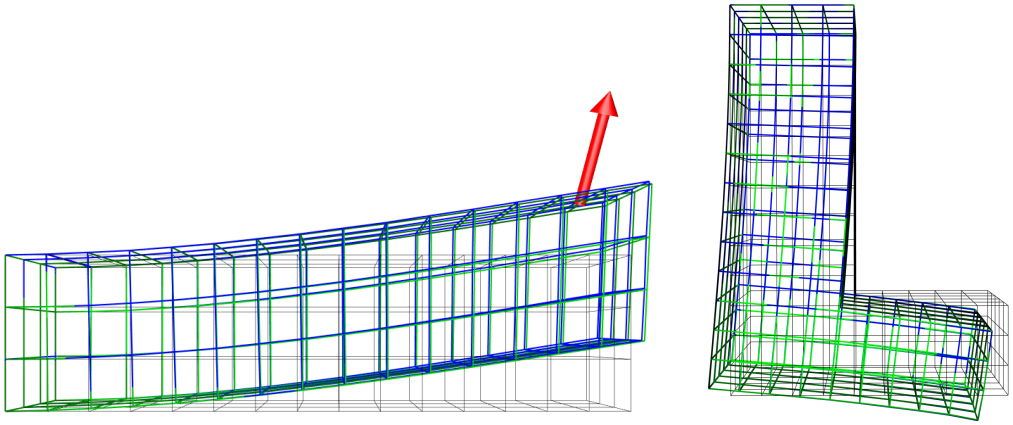


Figure 10.4: Comparaison de la prédiction U-Mesh (en vert) et de la solution EF (en bleu) pour une poutre encastrée et une lettre L.

Néanmoins, il faut noter que pour des maillages aussi grossier, les calculs EF pourraient être fait en temps-réel avec quelques optimizations. Pour montrer le potentiel de notre méthode, nous l'avons appliqué sur un maillage de poutre très fin pour lequel les EF ne peuvent pas satisfaire la contrainte de temps-réel. Les résultats montrent que notre méthode est capable de prédire les déformations volumiques d'un objet maillé très finement en 3 ms aussi ($\sim$ 100 fois plus rapide qu'une version optimisée des EF). Dans le prochain chapitre, nous allons voir comment étendre l'utilisation de U-Mesh à n'importequel type de géométrie ne coincidant pas nécessairement avec une structure régulière.

10.4 U-Mesh pour la chirurgie augmentée du foie

Comme cité précédemment, pour construire une réalité augmentée d'un organe, il est nécessaire de faire un recalage non rigide entre les images préopératoires (scan CT et modèle EF) et les images intraopératoires (nuage de points et vue de la surface de l'organe). Dans ce chapitre nous allons voir comment faire ce recallage élastique en utilisant U-Mesh, pour prédire la déformation volumique de l'organe à partir d'un nuage de points surfacique clairsemé.

Comme vu dans le chapitre précédent, pour que U-Mesh puisse faire des prédictions, il est impératif d'encoder le champ de déplacement dans une grille régulière. Pour ce faire, plusieurs options s'offrent à nous. La première consiste à mailler la géométrie de l'objet avec n'importequel type d'éléments (tétrahèdres par exemple) pour y réaliser les calculs EF. Ensuite, le maillage peut être plongé dans une grille régulière et

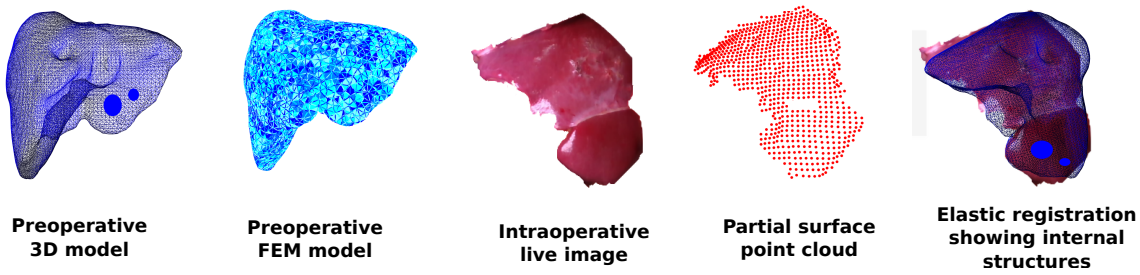


Figure 10.5: Étapes d'une chirurgie augmentée du foie.

les déplacements aux nœuds de la grille peuvent être obtenus par interpolation. Il faut se souvenir qu'avec cette approche on introduit une erreur d'approximation du volume non négligeable. Une autre alternative, plus précise, consiste à plonger la surface de l'objet dans une grille régulière et utiliser une méthode de type IBM (*immersed-boundary method* en anglais) pour construire un maillage hexahédrique régulier. Les cellules traversées par la frontière de l'objet sont subdivisées pour intégrer correctement le volume de l'objet.

La génération des données va être légèrement différente de celle du chapitre précédent puisque cette fois-ci on va enregistrer les couples (U_s, U_v) où U_s est le déplacement surfacique visible lors de la chirurgie et U_v est le déplacement volumique de l'organe. On applique des forces aléatoires sur la surface et on génère 2000 échantillons de foies déformés. La génération des données et l'entrainement du réseau doivent avoir lieu en un temps réduit puisqu'il peut parfois n'y avoir que quelques heures entre le moment où le scan CT préopératoire est acquis et le moment de l'intervention. Au total, les données ont été générées et le réseau entrainé en moins de 5 heures.

Une fois le réseau entrainé, nous avons fait des tests sur un foie synthétique puis sur des données réelles. Sur les données synthétiques, le réseau fait des prédictions avec moins de 0.05 mm d'*erreur nodale moyenne*. Les données réelles proviennent d'un foie humain ex vivo. Dix marqueurs sont injectés à l'intérieur du foie pour calculer des TRE (*target registration error* en anglais) et l'organe est déformé puis scanné pour avoir une vérité terrain à laquelle se comparer. On obtient un TRE moyen de 7.5 mm en faisant des predictions avec U-Mesh, ce qui est sensiblement la même chose que l'erreur obtenue avec les EF avec pour différence le temps de calcul (U-Mesh est 500 fois plus rapide que les EF). En effet, dans tous les scenarii les prédictions sont faites en seulement 3 ms. Dans les images ci-dessous sont illustrés le nuage de point RGB-D donné en entrée au réseau, puis la prediction du réseau (en blue turquoise). La prédiction peut être supperposée à la vue opératoire pour construire la réalité augmentée.

Pour faire ces prédictions, nous avons utilisé des propriétés mécaniques moyennes.

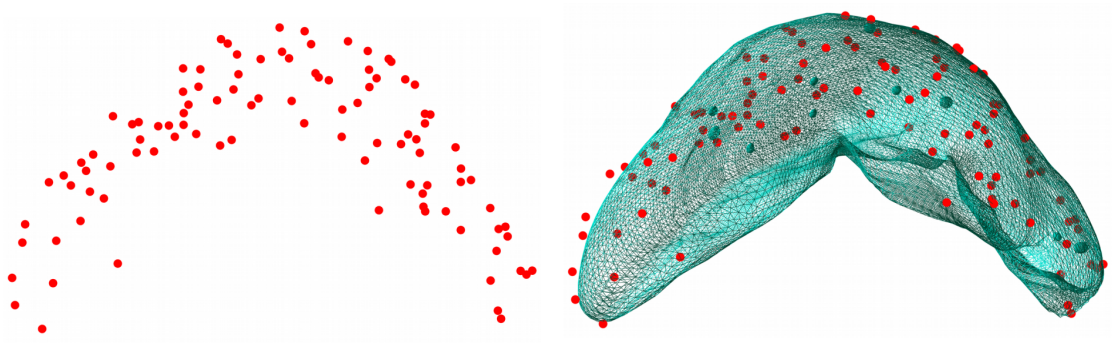


Figure 10.6: Nuage de points RGB-D donné en entrée au réseau et prédiction résultante.

En particulier, le foie est fixé au niveau du ligament falciforme dont l'emplacement n'est qu'approximatif. Aussi, nous avons choisi un module d'Young moyen (5000 Pa). Dans ce cas, puisque nous travaillons avec des déplacements de surface et que nous considérons le foie comme étant homogène, le module d'Young n'a pas d'impact sur la déformation résultante [Miller et al., 2013]. Cependant si on voulait appliquer des forces, une meilleure approximation du paramètre est recquise. Pour ce faire, nous proposons dans le dernier chapitre de la thèse, d'estimer les paramètres mécaniques et les conditions aux limites d'un foie en utilisant des filtres de Kalman. Ainsi, les simulations EF spécifiques au patient sont utilisées pour rafiner le réseau (préalablement entrainer sur un modèle moyen) en utilisant du *transfert de connaissance* (*transfer learning* en anglais).

10.5 U-Mesh pour la biopsie du sein

Lorsqu'il y a des lésions suspectes dans une mammographie, il est préférable de réaliser une biopsie afin de vérifier leur malignité. Pour guider l'aiguille vers la lésion, la biopsie est souvent guidée par des images ultrasons (US) [O'Flynn et al., 2010]. Cependant, il n'est pas toujours facile de suivre les lésions sur ces images. En effet, elles ne sont pas toujours visibles suite au contraste réduit des US, et de plus, la navigation 3D est faite en se basant sur des informations 2D. Puisque nous disposons du scan CT préopératoire avec la localisation exacte des lésions, trouver une méthode qui mettrait à jour la positions des lésions pendant l'intervention serait très bénéfique. Comme pour la chirurgie augmentée du foie, on peut faire un recalage élastique entre le modèle préopératoire et l'image US en utilisant la méthode des EF. Là aussi, le sein est un organe hautement déformable et donc des équations non-linéaires entrent en jeu, ce qui rend les calculs incompatibles avec le temps-réel. Dans [Han et al., 2013], les auteurs proposent d'utiliser une formulation TLED basée sur GPU permettant de

calculer les déformations du sein en 30 ms, ce qui ne satisfait pas encore la contrainte de temps-réel.

Dans ce chapitre on propose d'utiliser la plateforme U-Mesh pour prédire la déformation du sein en réponse aux forces de compression induite par la sonde US. Dans ce cas, l'entrée du réseau comporte les déplacements des nœuds en contact avec la sonde US. La méthode est validée sur des données issues d'un fantome du sein.

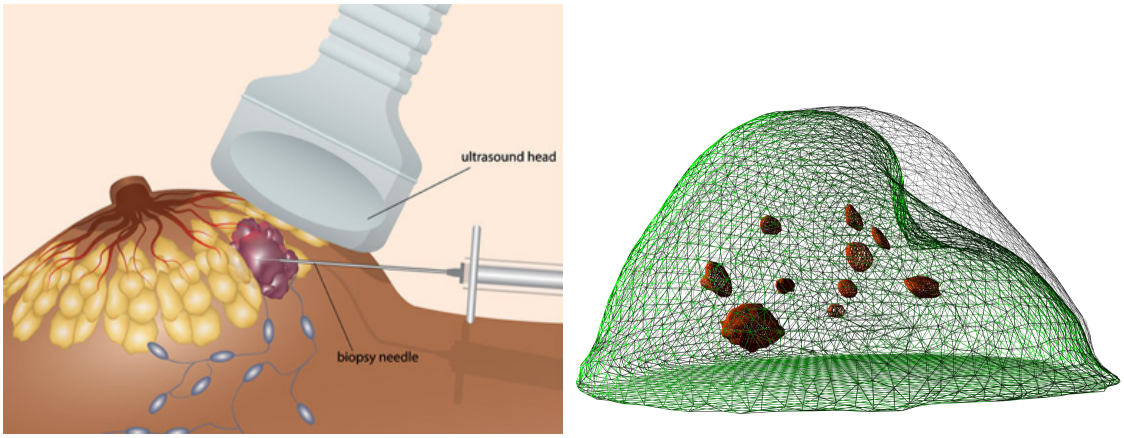


Figure 10.7: Prédiction U-Mesh en réponse à la compression induite par la sonde US.

Pour générer les données d'entrainement, on construit un modèle EF du sein. À partir du scan préopératoire on peut reconstruire la géométrie du fantome avec les lésions internes. On la plonge dans une grille régulière et on utilise une méthode IBM pour utiliser une grille hexahédrique pour nos simulations EF. Puisque l'entrée du réseau correspond à des déplacements surfaciques et qu'on considère le sein comme étant homogène, le module d'Young est arbitrairement initialisé. On génère 2000 échantillons de (U_s, U_v) où cette fois-ci, U_s correspond au déplacement des nœuds sous la sonde US. Pour ce faire, à chaque pas de simulation, on choisi un point p à la surface du sein, on choisit une boite orientée A centrée en p (A ayant la même dimension que la sonde US), et on applique une force normale à la surface sur tous les noeuds surfaciques contenus dans A.

Une fois que les données sont générées et que le réseau est entrainé ($\sim$ 5 heures), on prédit des déplacements sur des données synthétiques. On obtient des erreurs nodales moyennes de 0.05 mm ce qui montre bien encore une fois que le réseau a parfaitement appris le modèle avec lequel il a été entrainé. Dans un second temps, nous avons validé notre approche sur des données d'un fantome en sillicone contenant 10 lésions. On impose des déformations avec une sonde US qui est suivie dans l'espace. On connaît donc sa position et son orientation à chaque instant. Ainsi, on peut extraire l'emplacement des nœuds à la surface du sein en contact avec la sonde.

Pour chacune des 10 lésions, on place la sonde de telle sorte que la lésion considérée soit visible sur l'image US. Dans un premier temps, la sonde touche à peine le sein (donc aucune déformation n'est induite). Cette première étape permet de calibrer le système afin de référencer le modèle et la sonde dans un même repère. Ensuite on impose 4 déformations d'amplitudes croissantes. Ainsi on obtient 40 acquisitions avec des déformations variables. Lorsqu'on compare la prédiction U-Mesh avec la vérité terrain (obtenue à partir de l'image US), on obtient un TRE de 4.2 mm en moyenne. Encore une fois, on constate que U-Mesh fait aussi bien que la méthode EF utilisée pour l'entrainer.

En résumé nous avons construit une plateforme faisant l'interface entre des simulations EF précises mais couteuses en temps de calcul, et des applications nécessitant précision et rapidité simultanément.

10.6 Conclusion

Dans cette thèse nous avons démontré comment l'utilisation d'algorithmes d'apprentissage automatique en combinaison avec des méthodes numériques classiques, pouvait améliorer les interventions assistées par ordinateur. En particulier, nous avons présenté la méthode U-Mesh basée sur des réseaux de neurones pronfonds, qui modélise le comportement biomécanique d'un organe, tout en respectant les spécificités de chaque patient et en satisfaisant la contrainte de temps-réel inhérente à la chirurgie guidée par l'image.

Dans une phase préopératoire, on construit un modèle *éléments-finis* de l'organe considéré avec des paramètres moyens, afin de générer une base d'apprentissage pour entrainer le réseau. Ensuite, pendant la chirurgie, les propriétés mécaniques des organes sont identifiées avec de l'inférence Bayesienne et de nouvelles données patient-spécifiques peuvent être générées. Les poids des réseaux préalablement entrainés, peuvent être mis à jour par *transfert de connaissance* en utilisant les nouvelles données simulées.

Bibliography

[Abdel-Misih and Bloomston, 2010] Abdel-Misih, S. R., Bloomston, M., 2010. Liver anatomy. Surgical Clinics, 90(4), pp. 643-653.

[Aggarwal et al., 2018] Aggarwal, V., Asadi, H., Gupta, M., Lee, J. J., Yu, D., 2018. Covfefe: A Computer Vision Approach For Estimating Force Exertion. arXiv:1809.09293.

[Adagolodjo et al., 2016] Adagolodjo, Y., Goffin, L., de Mathelin, M., Courtecuisse, H., 2016. Inverse real-time Finite Element simulation for robotic control of flexible needle insertion in deformable tissues. IEEE/RSJ International Conference on Intelligent Robots and Systems.

[Allard et al., 2007] Allard, J., Cotin, S., Faure, F., Bensoussan, P. J., Poyer, F., Duriez, C., Delingette, H., Grisoni, L., 2007. SOFA - an open source framework for medical simulation. In MMVR 15-Medicine Meets Virtual Reality (125), pp. 13-18.

[Alvarez et al., 2018] Alvarez, P., Chabanas, M., Rouzé, S., Castro, M., Payan, Y., Dillenseger, J.L., 2018. Lung deformation between preoperative CT and intraoperative CBCT for thoracoscopic surgery: a case study. SPIE Medical Imaging.

[Amundarain et al., 2004] Amundarain, A., Borro, D., García-Alonso, A., Gil, J. J., Matey, L., Savall, J., 2004. Virtual reality for aircraft engines maintainability. Mechanics and Industry, 5(2), pp. 121-127.

[Apesteguía et al., 2011] Apesteguía, L., Pina, L.J., 2011. Ultrasound-guided core-needle biopsy of breast lesions. Insights into imaging, 2(4), pp. 493-500.

[Apostolopoulos et al., 2017] Apostolopoulos, S., Sznitman, R., 2017. Efficient OCT volume reconstruction from slitlamp microscopes. IEEE transactions on biomedical engineering, 64(10), pp. 2403-2410.

[Asejczyk-Widlicka et al., 2008] Asejczyk-Widlicka, M., Pierscionek, B. K., 2008. The elasticity and rigidity of the outer coats of the eye. British Journal of Ophthalmology, 92(10), pp. 1415-1418.

[Aviles et al., 2014] Aviles, A. I., Marban, A., Sobrevilla, P., Fernandez, J., Casals, A., 2014. A recurrent neural network approach for 3d vision-based force estimation. International Conference on Image Processing Theory, Tools and Applications (IPTA 2014), pp. 1-6. IEEE.

[Aviles et al., 2015] Aviles, A. I., Alsaleh, S., Sobrevilla, P., Casals, A. 2015. Sensorless force estimation using a neuro-vision-based approach for robotic-assisted surgery. In Neural Engineering (NER), 2015 7th International IEEE/EMBS Conference on, pp. 86-89. IEEE.

[Ayache et al., 2006] Cotin, S., Delingette, H., Ayache, N., 1999. Real-time elastic deformations of soft tissues for surgery simulation. IEEE transactions on Visualization and Computer Graphics, 5(1), pp. 62-73.

[Balci et al., 2010] Balci, Y., Basmak, H., Kocaturk, B. K., Sahin, A., Ozdamar, K., 2010. The importance of measuring intraocular pressure using a tonometer in order to estimate the postmortem interval. The American journal of forensic medicine and pathology, 31(2), pp. 151-155.

[Barbic and James, 2008] Barbic, J., James, D. L., 2008. Six-dof haptic rendering of contact between geometrically complex reduced deformable models. IEEE Transactions on Haptics, 1(1), pp. 39-52.

[Bhattacharjee and Matous, 2016] Bhattacharjee, S., Matous, K., 2016. A nonlinear manifold-based reduced order model for multiscale analysis of heterogeneous hyperelastic materials. Journal of Computational Physics, (313), pp. 635-653.

[Benzley et al., 1995] Benzley, S.E., Perry, E., Merkley, K., Clark, B., Sjaardema, G., 1995. A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis. 4th IMR, (17) pp. 179-191.

[Bosman et al., 2014] Bosman, J., Haouchine, N., Dequidt, J., Peterlik, I., Cotin, S., Duriez, C., 2014. The role of ligaments: Patient-specific or scenario-specific? International Symposium on Biomedical Simulation (ISBMS 2014).

[Boutin et al., 2017] Boutin, C., dell'Isola, F., Giorgio, I. and Placidi, L., 2017. Linear pantographic sheets: Asymptotic micro-macro models identification. Mathematics and Mechanics of Complex Systems, 5(2), pp. 127-162.

[Bro-Nielsen et al., 1996] BroNielsen, M., and Cotin, S., 1996. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In Computer graphics forum 3(15), pp. 57-66.

[Brunet et al., 2019] Brunet, J.N., Mendizabal, A., Petit, A., Golse, N., Vibert, E., Cotin, S., 2019. Physics-based Deep Neural Network for Augmented Reality during Liver Surgery. In International Conference on Medical image computing and

computer-assisted intervention (MICCAI 2019). (11768), pp. 137-145, Springer, Cham.

[Chakraborty, 2020] Chakraborty, S., 2020. Transfer learning based multi-fidelity physics informed deep neural network. ArXiv

[Cignoni et al., 2008] Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., Ranzuglia, G., 2008. MeshLab: an Open-Source Mesh Processing Tool. Eurographics Italian Chapter Conference.

[Cifuentes et al., 1992] Cifuentes, A., Kalbag, A., 1992. A performance study of tetrahedral and hexahedral elements in 3-D finite element structural analysis. Finite Finite Elements in Analysis and Design, 12, pp. 313-318.

[Clements et al., 2008] Clements, L. W., Chapman, W. C., Dawant, B. M., Galloway, R. L., and Miga, M. I., 2008. Robust surface registration using salient anatomical features for image-guided liver surgery: Algorithm and validation. Medical physics, 35(6), pp. 2528-2540.

[Collins et al., 2013] Collins, T., Pizarro, D., Bartoli, A., Canis, M. and Bourdel, N., 2013. Real-time wide-baseline registration of the uterus in monocular laparoscopic videos. In International Conference on Medical image computing and computer-assisted intervention (MICCAI 2013). Springer, Cham.

[Comas et al., 2008] Comas, O., Taylor, Z. A., Allard, J., Ourselin, S., Cotin, S., Passenger, J., 2008. Efficient nonlinear FEM for soft tissue modelling and its GPU implementation within the open source framework SOFA. In International Symposium on Biomedical Simulation, pp. 28-39. Springer, Cham.

[Courtecuisse et al., 2015] Courtecuisse, H., Adagolodjo, Y., Delingette, H., Duriez, C., 2015 Haptic Renderingof Hyperelastic Models with Friction. 2015 IEEE/RSJ International Conference on Intelligent Robotsand Systems (IROS), pp.591-596.

[Cybenko, 1989] Cybenko, G., 1989. Approximation by superpositions of a sigmoidal function. Math. Control Signal Systems 2, pp. 303-314.

[De Angelo et al., 2019] De Angelo, M., Barchiesi, E., Giorgio, I. and Abali B. E., 2019. Numerical identification of constitutive parameters in reduced order bi-dimensional models for pantographic structures: Application to out-of-plane buckling, Archive of Applied Mechanics, 89(7), pp. 1333-1358.

[Delingette et al., 2004] Delingette, H., and Ayache, N., 2004. Soft tissue modeling for surgery simulation. Handbook of Numerical Analysis, (12), pp. 453-550.

[Düster et al., 2008] Düster, A., Parvizianb, J., Yanga, Z., Ranka, E., 2008. The finite cell method for three-dimensional problems of solid mechanics. Computer Methods in Applied Mechanics and Engineering. 197(45-48), pp. 3768-3782.

[Duriez, 2013] Duriez, C., 2013. Control of Elastic Soft Robots based on Real-Time Finite Element Method. IEEE International Conference on Robotics and Automation (ICRA 2013).

[Duriez, 2013] Duriez, C., 2013. Real-time haptic simulation of medical procedures involving deformations and device-tissue interactions. Modeling and Simulation. Université des Sciences et Technologie de Lille.

[Fedorov et al., 2012] 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., Kikinis, R., 2012. 3D Slicer as an image computing platform for the Quantitative Imaging Network. Magnetic resonance imaging, 30(9), pp. 1323-1341.

[Felippa et al., 2005] Felippa, C.A., Haugen, B., 2005. A unified formulation of small-strain corotational finite elements: I. Theory. Computer Methods in Applied Mechanics and Engineering, 194(21-24), pp. 2285-2335.

[Fetene et al., 2018] Fetene, B., N., Rajkumar S., and Uday S., D., 2018. FEM-based neural network modeling of laser-assisted bending. Neural Computing and Applications 29(6), pp. 69-82.

[Gee et al., 2010] Gee M.W., Forster C., Wall W.A., 2010. A computational strategy for prestressing patient-specific biomechanical problems under finite deformation. International Journal for Numerical Methods inBiomedical Engineering. (26), pp. 52-72.

[Goury and Duriez, 2018] Goury, O., Duriez, C., 2018. Fast, Generic, and Reliable Control and Simulation of Soft Robots Using Model Order Reduction. IEEE Transactions on Robotics, (99), pp. 1-12.

[Guo et al., 2017] Guo, R., Lu, G., Qin, B., Fei, B., 2017. Ultrasound imaging technologies for breast cancer detection and management: A review. Ultrasound in medicine and biology.

[Haferssas et al., 2017] Haferssas, R., Jolivet, P., Nataf, F., 2017. An Additive Schwarz Method Type Theory for Lions's Algorithm and a Symmetrized Optimized Restricted Additive Schwarz Method. SIAM Journal on Scientific Computing, 39(4), pp. 1345-1365.

[Haidegger et al., 2017] Haidegger, T., Benyó, B., Kovács, L., Benyó, Z., 2009. Force sensing and force control for surgical robots. In 7th IFAC Symposium on Modeling and Control in Biomedical Systems, 7(1), pp. 413-418.

[Hamilton et al., 2008] Hamilton, K. E., Pye, D. C., 2008. Young's modulus in normal corneas and the effect on applanation tonometry. Optometry and Vision Science, 85(6), pp. 445-450.

[Han et al., 2013] Han, L., Hipwell, J.H., Eiben, B., Barratt, D., Modat, M., Ourselin, S., Hawkes, D.J., 2013. A nonlinear biomechanical model based registration method for aligning prone and supine MR breast images. IEEE transactions on medical imaging, 33(3), pp. 682-694.

[Haouchine et al., 2013a] Haouchine, N., Dequidt, J., Berger, M. O., Cotin, S., 2013. Deformation-based augmented reality for hepatic surgery. Studies in health technology and informatics, 184.

[Haouchine et al., 2013b] Haouchine, N., Dequidt, J., Peterlik, I., Kerrien, E., Berger, M., Cotin, S., 2013. Image-guided simulation of heterogeneous tissue deformation for augmented reality during hepatic surgery. IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2013), pp. 199-208.

[Haouchine et al., 2018] Haouchine, N., Kuang, W., Cotin, S., Yip, M. C., 2018. Vision-based Force Feedback Estimation for Robot-assisted Surgery using Instrument-constrained Biomechanical 3D Maps. IEEE Robotics and Automation Letters.

[Heiselman et al., 2017] Heiselman, J.S., Clements, L.W., Collins, J.A., Weis, J.A., Simpson, A.L., Geevarghese, S.K., Kingham, T.P., Jarnagin, W.R., Miga, M.I., 2017. Characterization and correction of intraoperative soft tissue deformation in image-guided laparoscopic liver surgery. Journal of Medical Imaging, 5(2).

[Hipwell et al., 2016] Hipwell, J.H., Vavourakis, V., Han, L., Mertzanidou, T., Eiben, B., Hawkes, D.J., 2016. A review of biomechanically informed breast image registration. Physics in Medicine & Biology, 61(2).

[Hornik, 1991] Hornik, K., 1991. Approximation capabilities of multilayer feedforward networks. Neural Networks 4(2), pp. 251-257.

[Hu et al., 2016] Hu, Y., Ahmed, H.U., Taylor, Z., Allen, C., Emberton, M., Hawkes, D., Barratt, D., 2012. MR to ultrasound registration for image-guided prostate interventions. Medical image analysis, 16(3), pp. 687-703.

[Jagtap et al., 2004] Jagtap, A. D., and Riviere, C. N., 2004. Applied force during vitreoretinal microsurgery with handheld instruments. In The 26th Annual Inter-

national Conference of the IEEE Engineering in Medicine and Biology Society, Vol. 1, pp. 2771-2773.

[Johnsen et al., 2015] Johnsen, S. F., Taylor, Z. A., Clarkson, M. J., Hipwell, J., Modat, M., Eiben, B., Han, L. , Hu, Y. , Mertzanidou, T. , Hawkes, D. J. Ourselin, S., 2015. NiftySim: A GPU-based nonlinear finite element package for simulation of soft tissue biomechanics. International journal of computer assisted radiology and surgery, 10(7), pp. 1077-1095.

[Joldes et al., 2010] Joldes, G.R., Wittek, A. and Miller, K., 2010. Real-time nonlinear finite element computations on GPU–Application to neurosurgical simulation. Computer methods in applied mechanics and engineering, 199(49-52), pp. 3305-3314.

[Julier et al., 1995] Julier, S. J., Uhlmann, J. K. and Durrant-Whyte, H. F., 1995. A new approach for filtering nonlinear systems, Proceedings of ACC'95 (3), pp. 1628-1632.

[Kingma and Ba, 2014] Kingma, D. P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv:1412.6980.

[Kourounis et al., 2018] Kourounis, D., Fuchs, A., Schenk, O., 2018. Toward the next generation of multiperiod optimal power flow solvers. IEEE Transactions on Power Systems, 33(4), pp. 4005-4014.

[Krücker et al., 2011] Krücker, J., Xu, S., Venkatesan, A., Locklin, J.K., Amalou, H., Glossop, N., Wood, B., 2011. Clinical utility of real-time fusion guidance for biopsy and ablation. Journal of Vascular Interventional Radiology, 22(4), pp. 515-524.

[Lasso et al., 2014] Lasso, A., Heffter, T., Rankin, A., Pinter, C., Ungi, T., Fichtinger, G., 2014. PLUS: open-source toolkit for ultrasound-guided intervention systems. IEEE Transactions on Biomedical Engineering, 61(10), pp. 2527-2537.

[Le Cun, 1988] Le Cun, Y., 1988. A theoretical framework for back-propagation. Proceedings of the 1988 Connectionist Models Summer School, pp. 21-28.

[Li et al., 2017] Li, H., Zheng, J., Cai, J. Y., Li, S. H., Zhang, J. B., Wang, X. M., Chen, G. H., Yang, Y., Wang, G. S., 2017. Laparoscopic vs open hepatectomy for hepatolithiasis: An updated systematic review and meta-analysis. World journal of gastroenterology, 23(43), pp. 7791-7806.

[Lorente et al., 2017] Lorente, D., Martínez-Martínez, F., Rupérez, M. J., Lago, M. A., Martínez-Sober, M., Escandell-Montero, P., Martínez-Martínez, J.M. , Martinez Sanchis, S., Serrano-López, A. , Monserrat, C, Martín-Guerrero, J. D., 2017. A framework for modelling the biomechanical behaviour of the human liver during

breathing in real time using machine learning. Expert Systems with Applications, 71, pp. 342-357.

[Lu et al., 2009] Lu J., Zhao X.F., 2009. Pointwise Identification of Elastic Properties in Nonlinear Hyperelastic Membranes-Part I: Theoretical and Computational Developments. Journal of Applied Mechanics. 76(6).

[Luo et al., 2018] Luo, R., Shao, T., Wang, H., Xu, W., Zhou, K., Yang, Y., 2018. DeepWarp: DNN-based Nonlinear Deformation. arXiv:1803.09109.

[Marchessau et al., 2010] Marchesseau, S., Heimann, T., Chatelin, S., Willinger, R., Delingette, H., 2010. Multiplicative jacobian energy decomposition method for fast porous visco-hyperelastic soft tissue model. In International Conference on Medical image computing and computer-assisted intervention (MICCAI 2010), pp. 235-242. Springer, Cham.

[Martínez et al., 2017] Martínez-Martínez, F., Rupérez-Moreno, M.J., Martínez-Sober, M., Solves-Llorens, J.A., Lorente, D., Serrano-López, A.J., Martínez-Sanchis, S., Monserrat, C., Martin-Guerrero, J.D., 2017. A finite element-based machine learning approach for modeling the mechanical behavior of the breast tissues under compression in real-time. Computers in biology and medicine, 90, pp. 116-124.

[Meenink et al., 2012] Meenink, H. C. M., Hendrix, R., Naus, G. J. L., Beelen, M. J., Nijmeijer, H., Steinbuch, M., van Oosterhout, E.J.G.M., de Smet, M. D., 2012. Robot-assisted vitreoretinal surgery. In Medical Robotics, pp. 185-209.

[Meier et al., 2005] Meier, U., López, O., Monserrat, C. U., Alcañiz J.M., 2005. Real-time deformable models for surgery simulation: a survey. Comput Methods Programs Biomed., 77(3), pp. 183-197.

[Meister et al., 2018] Meister, F., Passerini, T., Mihalef, V., Tuysuzoglu, A., Maier, A., Mansi, T., 2018. Towards Fast Biomechanical Modeling of Soft Tissue Using Neural Networks. arXiv:1812.06186 .

[Mendizabal et al., 2018] Mendizabal, A., Fountoukidou, T., Hermann, J., Sznitman, R., Cotin, S., 2018. A Combined Simulation and Machine Learning Approach for Image-Based Force Classification During Robotized Intravitreal Injections. In International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI 2018), pp. 12-20. Springer, Cham.

[Mendizabal et al., 2019] Mendizabal, A., Márquez-Neila, P., and Cotin, S. 2020. Simulation of hyperelastic materials in real-time using deep learning. Medical Image Analyses. Vol. 59.

[Mendizabal et al., 2019] Mendizabal, A., Tagliabue, E., Brunet, J-N., Dall'Alba, D., Fiorini, P., Cotin, S., 2019. Physics-based Deep Neural Network for Real-Time Lesion Tracking in Ultrasound-guided Breast Biopsy. Computational Biomechanics for Medicine Workshop.

[Mendizabal et al., 2019] Mendizabal, A., Sznitman, R., and Cotin, S., 2019. Force classification during robotic interventions through simulation-trained neural networks. International journal of computer assisted radiology and surgery, 14(9), pp. 1601-1610.

[Messina et al., 2002] Messina, L.M., Schneider, D.B., Chuter, T.A., et al., 2002. Integrated fellowship in vascular surgery and intervention radiology: a new paradigm in vascular training. Ann Surg. 236(4). pp. 408-415.

[Miller et al., 2007] Miller, K., Joldes, G., Lance, D., Wittek, A., 2007. Total Lagrangian explicit dynamics finite element algorithm for computing soft tissue deformation. Communications in numerical methods in engineering, 23(2), pp. 121-134.

[Miller et al., 2013] Miller, K., Lu, J., 2013. On the prospect of patient-specific biomechanics without patient-specific properties of tissues. Journal of the mechanical behavior of biomedical materials, 27, pp. 154-166.

[Modrzejewski et al., 2018] Modrzejewski, R., Collins, T., Bartoli, A., Hostettler, A., Marescaux, J., 2018. Soft-body registration of pre-operative 3d models to intra-operative RGBD partial body scans. In International Conference on Medical image computing and computer-assisted intervention (MICCAI 2018), pp. 39-46. Springer, Cham.

[Moireau et al., 2011] Moireau, P., Chapelle, D., 2011. Reduced-order Unscented Kalman Filtering with application to parameter identification in large-dimensional systems. ESAIM: Control, Optimisation and Calculus of Variations, 17(2), pp. 380-405.

[Morooka et al., 2008] Morooka, K. I., Chen, X., Kurazume, R., Uchida, S., Hara, K., Iwashita, Y., and Hashizume, M., 2008. Real-time nonlinear FEM with neural network for simulating soft organ model deformation. In International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI 2008), pp. 742-749. Springer, Cham.

[Mura et al., 2016] Mura, M., Abu-Kheil, Y., Ciuti, G., Visentini-Scarzanella, M., Menciassi, A., Dario, P., Dias, J., Seneviratne, L., 2016. Vision-based haptic feedback for capsule endoscopy navigation: a proof of concept. Journal of Micro-Bio Robotics, 11(1-4), pp. 35-45.

[Muthupillai et al., 1995] Muthupillai R., Lomas D.J., Rossman P.J., Greenleaf J.F., Manduca A., Ehman R.L., 1995. Magnetic-resonance elastography by direct visualization of propagating acoustic strain waves. Science. 269, pp. 1854-1857.

[Nikolaev et al., 2018] Nikolaev, S., Peterlík, I., and Cotin, S., 2018. Stochastic Correction of Boundary Conditions during Liver Surgery. In 2018 Colour and Visual Computing Symposium (CVCS) pp. 1-4. IEEE.

[Niroomandi et al., 2008] Niroomandi, S., Alfaro, I., Cueto, E., Chinesta, F., 2008. Real-time deformable models of non-linear tissues by model reduction techniques. Computer methods and programs in biomedicine, 91(3), pp. 223-231.

[Niroomandi et al., 2013] Niroomandi, S., Gonzalez, D., Alfaro, I., Bordeu, F., Leygue, A., Cueto, E., Chinesta, F., 2013. Real-time simulation of biological soft tissues: a PGD approach. International journal for numerical methods in biomedical engineering, 29(5), pp. 586-600.

[Niroomandi et al., 2017] Niroomandi, S., Alfaro, I., Cueto, E., Chinesta, F., 2010. Model order reduction for hyperelastic materials. International Journal for Numerical Methods in Engineering, 81(9), pp. 1180-1206.

[O'Flynn et al., 2010] O'Flynn, E.A.M., Wilson, A.R.M., Michell, M.J., 2010. Image-guided breast biopsy: state-of-the-art. Clinical radiology, 65(4), pp. 259-270.

[Olsen et al., 2002] Olsen, T. W., Sanderson, S., Feng, X., Hubbard, W. C., 2002. Porcine sclera: thickness and surface area. Investigative ophthalmology & visual science, 43(8), pp. 2529-2532.

[Pakhomov et al., 2017] Pakhomov, D., Premachandran, V., Allan, M., Azizian, M., Navab, N., 2017. Deep residual learning for instrument segmentation in robotic surgery. arXiv:1703.08580.

[Paulus et al., 2017] Paulus, C. J., Maier, R., Peterseim, D., Cotin, S., 2017. An Immersed Boundary Method for Detail-Preserving Soft Tissue Simulation from Medical Images. In International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI 2017), pp. 55-67. Springer, Cham

[Pellicer-Valero et al., 2019] Pellicer-Valero, O. J., Rupérez, M. J., Martínez-Sanchis, S., and Martín-Guerrero, J. D., 2019. Real-time biomechanical modeling of the liver using Machine Learning models trained on Finite Element Method simulations. Expert Systems with Applications, 113083.

[Peterlík et al., 2012] Peterlík, I., Duriez, C., Cotin, S., 2012. Modeling and real-time simulation of a vascularized liver tissue. In International Conference on Medical

image computing and computer-assisted intervention (MICCAI 2012), pp. 50-57. Springer, Cham

[Peterlík et al., 2014] Peterlík, I., Courtecuisse, H., Duriez, C., Cotin, S., 2014. Model-based identification of anatomical boundary conditions in living tissues. IPCAI. pp. 196-205

[Peterlík et al., 2017] Peterlík, I., Haouchine, N., Ručka, L., and Cotin, S., 2017. Image-driven stochastic identification of boundary conditions for predictive simulation. In International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI 2017) pp. 548-556. Springer, Cham.

[Petit et al., 2015] Petit, A., Lippiello, V., Siciliano, B., 2015. Real-time tracking of 3D elastic objects with an RGB-D sensor. IROS, pp. 3914-3921.

[Petit et al., 2018] Petit, A., Cotin, S., 2018. Environment-aware non-rigid registration in surgery using physics-based simulation. ACCV - 14th Asian Conference on Computer Vision.

[Pfeiffer et al., 2019] Pfeiffer, M., Riediger, C., Weitz, J., Speidel, S., 2019. Learning soft tissue behavior of organs for surgical navigation with convolutional neural networks. IJCARS pp. 1-9.

[Pfeiffer et al., 2020] Pfeiffer, M., Riediger, C., Leger, S., Kühn, J-P., Seppelt, D., Hoffmann, R-T., Weitz, J., Speidel, S., 2020. Non-Rigid Volume to Surface Registration using a Data-Driven Biomechanical Model. In International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI 2020). Springer, Cham.

[Plantefève et al., 2014] Plantefeve, R., Peterlik, I., Courtecuisse, H., Trivisonne, R., Radoux, J.P, Cotin, S., 2014. Atlas-Based Transfer of Boundary Conditions for Biomechanical Simulation. In International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI 2014), vol 17, pp-33-40. Springer, Cham.

[Plantefeve et al., 2016] Plantefeve, R., Peterlík, I., Haouchine, N., Cotin, S., 2016. Patient-specific biomechanical modeling for guidance during minimally-invasive hepatic surgery. Ann Biomed Eng. 44 (1), pp. 139-153.

[Raissi et al., 2019] Raissi, M., Perdikaris, P., Karniadakis, G.E., 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378, pp. 686-707.

[Rechowicz et al., 2013] Rechowicz, K. J., and McKenzie, F. D., 2013. Development and validation methodology of the Nuss procedure surgical planner. Simulation, 89(12), 1474-1488.

[Roewer-Despres et al., 2018] Roewer-Despres, F., Khan, N., Stavness, I., 2018. Towards finite element simulation using deep learning, 15th International Symposium on Computer Methods in Biomechanics and Biomedical Engineering.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention, pp. 234-241.

[Rosser et al., 2000] Rosser, J.C. Jr, Murayama, M., Gabriel, N.H., 2000. Minimally invasive surgical training solutions for the twenty-first century. Surg Clin North Am. 80(5). pp. 1607-1624.

[Ruiter et al., 2006] Ruiter, N., Stotzka, R., Muller, T.-O, Gemmeke, H., Reichenbach, J., Kaiser, W., 2006. Model-based registration of X-ray mammograms and MR images of the female breast. IEEE Transactions on Nuclear Science (53), pp. 204 - 211.

[Runge et al., 2017] Runge, G., Wiese, M.,Raatz, A., 2017. FEM-based training of artificial neural networks for modular soft robots. IEEE International Conference on Robotics and Biomimetics (ROBIO 2017).

[Ryckelynck, 2005] Ryckelynck, D., 2005. A priori hyperreduction method: an adaptive approach. Journal of computational physics, 202(1), pp. 346-366.

[Sarvazyan et al., 1998] Sarvazyan A.P., Rudenko O.V., Swanson S.D., Fowlkes J.B., Emelianov S.Y., 1998. Shear wave elasticityimaging: A new ultrasonic technology of medical diagnostics. Ultrasound in Medicine andBiology. 24, pp. 1419-1435.

[Shepherd et al., 2008] Shepherd, J. F., and Johnson, C. R., 2008. Hexahedral mesh generation constraints. Engineering with Computers, 24(3), pp. 195-213.

[Shewchuk, 1994] Shewchuk, J. R., 1994. An introduction to the conjugate gradient method without the agonizing pain.

[Sinkus et al., 2010] Sinkus, R., Daire, J.L., Van Beers, B.E., Vilgrain, V., 2010. Elasticity reconstruction: Beyond the assumption oflocal homogeneity. Comptes Rendus Mécanique. 338(7-8), pp. 474-479.

[Slawinski, 2007] Slawinski, M., 2007. Waves and rays in elastic continua. World Scientific.

[Suwelack et al., 2014] 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., Speidel, S., 2014. Physics-based shape matching for intraoperative image guidance. Med Phys. 41(11).

[Tagliabue et al., 2019] Tagliabue, E., Dall'Alba, D., Magnabosco, E., Tenga, C., Peterlik, I., Fiorini, P., 2019. Position-based modeling of lesion displacement in ultrasound-guided breast biopsy. IJCARS

[Tonutti et al., 2017] Tonutti, M., Gauthier G., and Guang-Zhong Y., 2017. A machine learning approach for real-time modelling of tissue deformation in image-guided neurosurgery. Artificial intelligence in medicine, 80, pp. 39-47.

[Troccaz, 2012] Troccaz, J., 2012. Medical robotics. ISTE Ltd.

[Ullrich et al., 2016] Ullrich, F., Michels, S., Lehmann, D., Pieters, R. S., Becker, M., Nelson, B. J., 2016. Assistive Device for Efficient Intravitreal Injections. Ophthalmic Surgery, Lasers and Imaging Retina, 47(8), pp. 752-762.

[Visentin et al., 2018] Visentin, F., Groenhuis, V., Maris, B., Dall'Alba, D., Siepel, F., Stramigioli, S., Fiorini, P., 2018. Iterative simulations to estimate the elastic properties from a series of mri images followed by mri-us validation. Medical and biological engineering and computing, 194(21-24), pp. 1-12.

[Wang et al., 2004] Wang, E., Nelson, T., Rauch, R., 2004. Back to elements-tetrahedra vs. hexahedra. Proceedings of the 2004 international ANSYS conference.

[Weber et al., 2017] Weber, S., Gavaghan, K., Wimmer, W., Williamson, T., Gerber, N., Anso, J., Bell, B., Feldmann, A., Rathgeb, C., Matulic, M., Stebinger, M.,Schneider, D., Mantokoudis, G., Scheidegger, O., Wagner, F., Kompis, M., Caversaccio, M., 2017. Instrument flight to the inner ear. Science robotics, 2(4).

[Xu et al., 2007] Xu, L., Lin, Y., Han, J.C., Xi, Z.N., Shen, H., Gao, P.Y., 2007. Magnetic resonance elastography of brain tumors:Preliminary results. Acta Radiologica, 48, pp. 327-330.

[Yushkevich et al., 2006] Yushkevich, P.A., Piven, J., Hazlett, H.C., Smith, R.G., Ho, S., Gee, J.C., Gerig, G., 2006. User-Guided 3D Active Contour Segmentation of Anatomical Structures: Significantly Improved Efficiency and Reliability. Neuroimage, 31(3), pp. 1116-1128.

[Zhao et al., 2009] Zhao, X.F., Chen, X.L., Lu, J., 2009. Pointwise Identification of Elastic Properties in Nonlinear HyperelasticMembranes-Part II: Experimental Validation. J Appl Mech-T Asme. 76, pp. 61011-61014.

[Zhou et al., 2019] Zhou, T., Ruan, S., Canu, S., 2019. A review: Deep learning for medical image segmentation using multi-modality fusion. Array, Vol. 3-4.

Andrea MENDIZABAL

Combinaison entre simulation numérique et apprentissage automatique

Résumé

Il existe une multitude de domaines en ingénierie nécessitant le calcul de déformations non-linéaires en temps réel, notamment dans le domaine de la médecine, avec les simulateurs pour l'entrainement des chirurgiens ou bien la chirurgie guidée par l'image. Dans un contexte de chirurgie augmentée, il est nécessaire de réaliser un recalage élastique des données préopératoires sur la vue intraopératoire de l'organe. L'objectif est de superposer en temps réel, un modèle virtuel spécifique à chaque patient, à la vue du champ opératoire dans le but de visualiser les structures internes de l'organe (tumeurs ou artères par exemple). Afin d'obtenir une précision adaptée, il est nécessaire de construire un modèle biomécanique personnalisé de l'organe. Pour ce faire, la méthode des éléments-finis est la technique préférée afin de prédire la déformation des tissus mous. Cependant la complexité des calculs impliqués (notamment dans le cas de déformations non-linéaires) rend cette méthode incompatible avec les exigences de temps réel et de précision, inhérentes au domaine d'application visé.

Pour répondre à ces contraintes, on propose de combiner des simulations éléments-finis avec des réseaux de neurones profonds pour modéliser le comportement biomécanique des tissus humains. En particulier on présente la plateforme U-Mesh permettant de prédire en temps-réel les déformations d'un organe comme le foie avec une précision adaptée à la chirurgie augmentée.

Mots clefs : Simulation temps-réel ; Simulation patient-spécifique ; Réseau de neurones pronfond ; Méthode des éléments-finis ; Chirurgie augmentée

Résumé en anglais

Many engineering applications require accurate numerical simulations of non-linear structures in real-time. Some important examples can be found in the field of medicine, in order to develop surgical training systems, or in the field of surgical navigation where augmented reality can bring significant improvements to the clinical gesture. To guarantee the accuracy of the simulations, patient-specific modeling must be pursued by taking into account personalized material parameters and boundary conditions. In the context of augmented surgery for instance, it is essential to perform an elastic registration between the preoperative and the intraoperative images. To this end, a patient-specific biomechanical model must be built to produce real-time finite-element simulations of the deformed organ. This is in practice very difficult to achieve as the problems to be solved are highly complex, in particular when non-linear deformations are considered.

In this work, we propose a method combining finite-element simulations and deep neural networks in order to satisfy the rapidity and accuracy requirements of medical applications. In particular, we present the U-Mesh framework, capable of predicting in real-time the shape of a highly deformable organ like the liver in order to guide surgeons during interventions where following the organ's deformation is crucial for the surgery to be successful.

Keywords: Real-time; Deep neural networks; Finite element method; Data-driven simulation; Augmented surgery