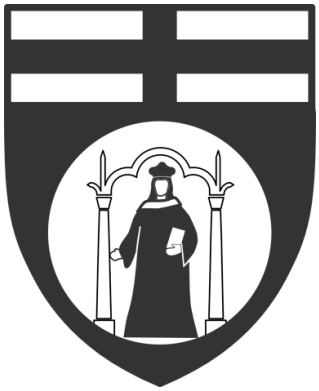
UNIVERSITA DEGLI STUDI DI GENOVA
POLYTECHNICAL SCHOOL - DIBRIS



Master of Science in Robotics Engineering

Estimation of soft tissue deformation for robot-assisted surgery

Candidate:

Riccardo Lastrico

Supervisors:

Prof. Fulvio Mastrogiovanni
Prof. Paolo Traverso
Dott. Alessandro Carfi

October 28, 2021

Abstract

With the technological advancement of the last 20 years, the robotic surgery has become more and more relevant and advanced, incorporating the new developments from the research to make it more supportive of the surgeon and being able to give him more information possible, both before and during the surgery.
The advent of augmented surgery would make possible to display holograms on the camera view from the robot, for instance a 3D model of the organ that is under surgery (and the surrounding ones) that deforms while being manipulated. A simulation has to be performed to keep track on how the virtual soft tissues will deform while being subjected to the forces that are applied to the real one.
Unfortunately, perform such a simulation is still a complex task for the mainstream computers to be computed in real time; from that reason, comes the idea of exploring the possibility of building a prediction model to avoid the necessity of running the simulation in the real time while the surgery is ongoing and still obtain information about the deformation of the soft tissue. To do so, it has been necessary to build a simulation to generate a large amount of data that would then used to train the model, a Neural Network; the network would be then used instead of the simulation in the loop, being much faster to execute with respect to the simulation, to fulfill the real time requirement that such application would have.
The results are promising and this work will serve as base for future projects and improvements.

Contents

List of figures **iv**

List of tables **vii**

1 Introduction **1**

2 Robotic surgery and personalized medicine **4**

2.1 Robotics surgery 4

2.1.1 Comparison of the outcome of robotic surgery with Open surgery and traditional MIS 6

2.1.2 The robot (daVinci Surgical System) 9

2.2 Robot-assisted surgery in Urology 12

2.2.1 Robot-assisted partial neprectomy (RAPN) 12

2.3 Limitation to robotic surgery 14

2.4 The future of robotic surgery 15

2.4.1 Future technological improvements 16

2.5 Augmented surgery 17

2.5.1 Augmented reality in hurology 18

3 Modeling the deformation behavior of soft tissues **20**

3.1 Material's properties 20

3.1.1 Constitutive Models 22

3.1.2 Heuristic (or particle) based models 25

3.1.3 Hybrid approaches 27

3.1.4 Extract material paramteres 27
3.2 Physical models 29
3.2.1 Obtain physical model of an organ from a patient scan . . 29
3.2.2 3D Model building pipeline 30

4 Simulation and simulation platform 33
4.1 Simulation platform 33
4.2 Components in SOFA 35
4.2.1 Models in SOFA 35
4.3 Other components 37
4.4 Simulation implementations examples 37
4.5 Anatomy of a basic simulation 39

5 Setting up the Simulation 45
5.1 Simulating the blood vessels' deformation 46
5.2 Simulating the kidney's Local deformation 52

6 The software development 55
6.1 The software architecture 56
6.1.1 Plot the state during the simulation 58
6.1.2 Data Parsing and analysis 59
6.2 Kidney's blood vessels deformation simulation 59
6.3 Kidney's local deformation simulation 61

7 Prediction model 62
7.1 Neural networks 62
7.2 Preparation of the data 64
7.3 Neural Networks in Python 65
7.4 Generating the data 66
7.4.1 Training sets 67
7.4.2 Test sets 68
7.5 Training and inference pipeline 68
7.6 Build the models 69

7.6.1 Regression Neural Network 69
7.6.2 LSTM Neural Network 71

8 Analysis of the results **75**
8.1 The monitored points 77
8.2 Comparing the models using metrics 82
8.2.1 Analysis on the test set '0927221920', veins 83
8.2.2 Summary table for the veins datasets 85
8.2.3 Summary table for the arteries datasets 85
8.3 Concluding remarks on the models 86

9 Conclusion **87**

A Additional graphs **92**

References **99**

List of Figures

2.1 Traditional MIS instrument 5
2.2 Robotic surgery vs. OP and MIS (From Tan et al. (2016)) 7
2.3 The complete daVinci robot system 9
2.4 The daVinci robot from Intuitive Surgical 10
2.5 Console, controls and instruments of the daVinci robot 12
2.6 Augmented surgery application with a kidney (Kong et al. (2016)) 18

3.1 Representation of a FEM force field in SOFA 24
3.2 Representation of a MSM in SOFA 26
3.3 3D model building pipeline 30
3.4 3D model of the kidney and blood vessels 31
3.5 3D model of the veins 32
3.6 3D model of the arteries 32

4.1 User interface of SOFA 34
4.2 FEM model with a tethrahedral shape 35
4.3 Collision model and its related bounding boxes 36
4.4 Mapping points (in yellow) 38
4.5 Example of a graph defining the scene of a basic simulation 39
4.6 Example of an object in a simulation 41
4.7 Example of an object in a simulation (focus on visual and collision model) 43

5.1 Graph of the blood vessels' simulation 47

5.2 Blood vessel's simulation 48
5.3 The poins that are being monitored, in helical shape 49
5.4 Applied constraints (in pink) 50
5.5 The kidney has been rotated of around 90° on the x axis 51
5.6 Graph of the local deformation simulation 52
5.7 Interaction between the kidney and the sphere from two different point of view 54

6.1 Scheme of the complete architecture 57
6.2 Example of graphs plotted during the simulation 59

7.1 An artificial neuron compared with an human one 63
7.2 Training and inference pipeline of the Neural Networks 68
7.3 Scheme of the implemented regression neural network 70
7.4 Moving window as input of the LSTM network 72
7.5 Scheme of the implemented LSTM network 73

8.1 Comparing the prediction of the models with the simulation output 76
8.2 Monitored points for the *veins* 77
8.3 Classification of the monitored points for the *veins* 78
8.4 Prediction of Point #11 using the Regression model (left) and the LSTM (right) 79
8.5 Prediction of Point #12 using the Regression model (left) and the LSTM (right) 80
8.6 Prediction of Point #31 using the Regression model (left) and the LSTM (right) 80
8.7 Prediction of Point #32 using the Regression model (left) and the LSTM (right) 81
8.8 Prediction of Point #38 using the Regression model (left) and the LSTM (right) 81
8.9 Prediction of Point #50 using the Regression model (left) and the LSTM (right) 82

9.1 Sw architecture to train the model 88
9.2 Sw architecture to inference from the model 89
9.3 Predicting in real time the shape of the blood vessels 90

A.1 Output of the Regression model compared with the original sequence (Dataset '027221920', veins) 93
A.2 Output of the Regression model compared with the original sequence (Dataset '027221920', veins) 94
A.3 Output of the Regression model compared with the original sequence (Dataset '027221920', veins) 95
A.4 Output of the LSTM model compared with the original sequence (Dataset '027221920', veins) 96
A.5 Output of the LSTM model compared with the original sequence (Dataset '027221920', veins) 97
A.6 Output of the LSTM model compared with the original sequence (Dataset '027221920', veins) 98

List of Tables

7.1 Training datasets 67
7.2 Test datasets 68

8.1 Comparing the distance from the points in the simulation to the predicted ones between the Regression model and the LSTM . . . 84
8.2 Metrics of the vein's datasets 85
8.3 Metrics of the arteries' datasets 85

Chapter 1

Introduction

The robotic surgery has become a topic of increasing interest in the last years due to the recent technological advances and the constant decreasing cost. Robots such as he daVinci from intuitive Surgical allows the surgeon to perform some kind of operation that wouldn't otherwise be possible.
Some surgeons have highlighted an important problem: the view from the camera during the operation can become unclear due to blood and the surgeon could lose visual contact with the organ on which the surgery is ongoing while it is being moved from the robotic instruments. This problem is even more relevant when dealing with organs such as a kidney, the one that I will take in account: its surface in a close-up view looks very homogeneous with very few distinguishable visual features, especially with a restricted view.
To try and solve this problem, there is currently ongoing research in the university labs regarding the tracking of human organs using inertial sensors, such as accelerometers and gyroscopes, to track the actual position and orientation of the organ while the surgery is going on. To get those feature, it would also be possible to apply visual markers on the surface of the organ and then estimate the 3D pose from the camera view.
By having some reliable feature about the organ's position and rotation, it would be possible to build an application of augmented surgery: that would allow to superimpose on the camera video flow that comes from a surgical robot a 3D

hologram of the human organ on the actual organ, together with its appendices; in the case of the kidney, for instance, the blood vessels or the urethra.
The augmented reality applied to a surgery can in fact be very helpful not only to represent the organ itself, but to give contextual information about what is going on in the region of the body that the surgeon is operating.
The blood vessel connected to the kidney are subjected to a lot of forces and deformation while the surgeon is manipulating the organ: study how they behave in those situation, how they deform or even if some deformation can become problematic and give information about them can be a very useful tool for the surgeon.
A surgical simulation can be useful for the surgeon itself also to be performed before the actual surgery, to see which are the motion that can be problematic or even dangerous, using a personalized model based on the patient's organ. Obviously, we cannot attach a large amount of sensors along the surface of the blood vessels to track how their shape changes while being manipulated.
From that came the main idea for the thesis: build a simulation that could simulate the deformation of the blood vessel depending on the motion of the kidney they're attached to and from that be able to extract data that can be analyzed in real time while the simulation is going on, by showing some plots on how the soft tissues are affected by forces, how their position will evolve over time or even more qualitative information (to detect and advise the surgeon about possible dangerous situation), based on the available features.
Unfortunately, such a simulation is not yet able to run in real time during the surgery, being computationally intensive, using the position and rotation data from the actual kidney while the surgery is going on. For this reason, we decided to try and see if it could be possible to build a model from the data generated in the simulation about the deformation of the blood vessels depending on the position assumed by the simulated kidney to predict the evolution of the position of veins and arteries.
By using Neural Networks, a very powerful instrument, we can train the model with the data recorded while the simulation running and then be able to predict the deformation and the motion of the blood vessels by just specifying the pose

of the kidney.
The simulation platform chosen is SOFA (Inria), being very focused toward medical simulation and very flexible in terms of control and data recording. The model of the kidney and blood vessels used in the simulation comes from a CT of the actual patient: that would make the application very precise because the 3D model that generates the data and with the surgeon can interact with is actually true to reality.
It has been necessary to perform a literature study about the techniques to simulate soft tissues to choose the most appropriate one to use, in addition to the new findings regarding robotic surgery and the techniques to build a 3D model of the patient's organ.
I developed many software instruments, from the simulation and architecture around it, necessary to send controls, log the state of the simulation and visualize the data in real time (while the simulation was running) to the components to prepare the data for the use of Neural Netwoks and the implementation of the latter and, additionally, the instruments to validate the results coming from the from the NN and compare them to the output of the simulation.

Chapter 2

Robotic surgery and personalized medicine

2.1 Robotics surgery

The field of robotic surgery has born in a period of time where minimal invasive surgical (MIS) technologies were increasingly being adopted by surgeons, to enhance their outcomes; for this reason, there was an increasing demand for safer operations and greater surgical precision.

The MIS approaches (such as laparoscopy) have different benefits with respect to the traditional open surgery; reduced wound access trauma, shorter hospital stay, improved visualization, less postoperative complication and disfigurement (Ashrafian et al. (2017)). The less tissue trauma and a speedier discharge would, as a consequence, increase cost-efficiency of the surgery. There were still some limitation associated with the laparoscopic procedures: the lack of tactile feedback, the 2D view, and the limited degrees of freedom of the laparoscopic instruments. The first robots for surgery were designed to outperform hand biopsies in terms of accuracy and surgical precision, as the natural evolution to laparoscopic surgery. Work with those robot suggested that a programmable computer linked to an high precision surgical cutting device could offer higher levels of operative accuracy when compared with conventional surgical methods. Robots could poten-

Figure 2.1: Traditional MIS instrument

tially offer more than an equivalent of an open surgery but with smaller incisions, an operation with a robot would allow a higher level of tissue discrimination, dissection and repair. They have advantages in reducing the systemic inflammatory while providing improved precision and accuracy in surgical technique because of superior 3D dexterity; in addition it offers potential in future developments including digitally enhanced analysis of tissues and .

The first surgical robots have been introduced more than thirty years ago, from that time surgical robots have become more and more relevant due to the technical benefits of modern robotic platforms. For the surgeon side, technical advantages the potential for better visualisation (higher magnification) with stereoscopic views, elimination of hand tremor allowing greater precision and improved dexterity as a result of the "robotic wrist".

To improve the kinematics,large external movements of the surgical hands can be scaled down and transformed to limited internal movements of the "robotic hands"; this allow to improves ergonomics and extend the surgical ability to perform complex technical tasks in a limited space. The surgeon is able to work in an ergonomic environment with less stress, achieving higher levels of concentration;

there may also be less need for assistance once surgery is under way.
The computerized nature of the surgical robot allows integration of real-time and previously recorded data utilisation, so that it could accommodate complex intra-operative factors such as compensating for the beating movement of the heart, making it unnecessary to stop the heart during cardiothoracic surgery (Camarillo et al. (2004), Ashrafian et al. (2017)).
Current robotic surgical evidence points towards a convincing reduction in post-operative surgical and non-surgical complications, reduced blood loss, improved recovery rates, improved cosmesis and reduced length of stay in comparison with open surgery; The comparison with MIS however is equivocal, although several studies do show some advantages in length of stay, conversion rate and estimated blood loss.
In terms of specialist surgeries, the benefits of robotic techniques have enabled increasingly complex procedures especially in the field of urology: Using an open approach would be extremely invasive, but using a robotic approach there is potential for a return to full physical fitness in brief time (Pearce et al. (2017)); these are outcome markers that can demonstrate the incremental gains offered by robotic surgery.

2.1.1 Comparison of the outcome of robotic surgery with Open surgery and traditional MIS

Tan et al. (2016) performed a systematic review of the literature for the first 30 years of robotic surgery (1985–2015); they identified more then 100 studies on around 15000 patients.
The studies that compares robotic vs. open surgery (OS), demonstrates lower blood loss at 50.5%, lower transfusion rate at 27.2%, lower length of hospital stay at 69.5%, and reduction of 30-day overall complication rate at 63.7% in favour of robotic surgery.
For robotic surgery vs traditional MIS, the studies demonstrated mildly reduced blood loss at 85.3% and transfusion rate at 62.1% in favour of robotic surgery

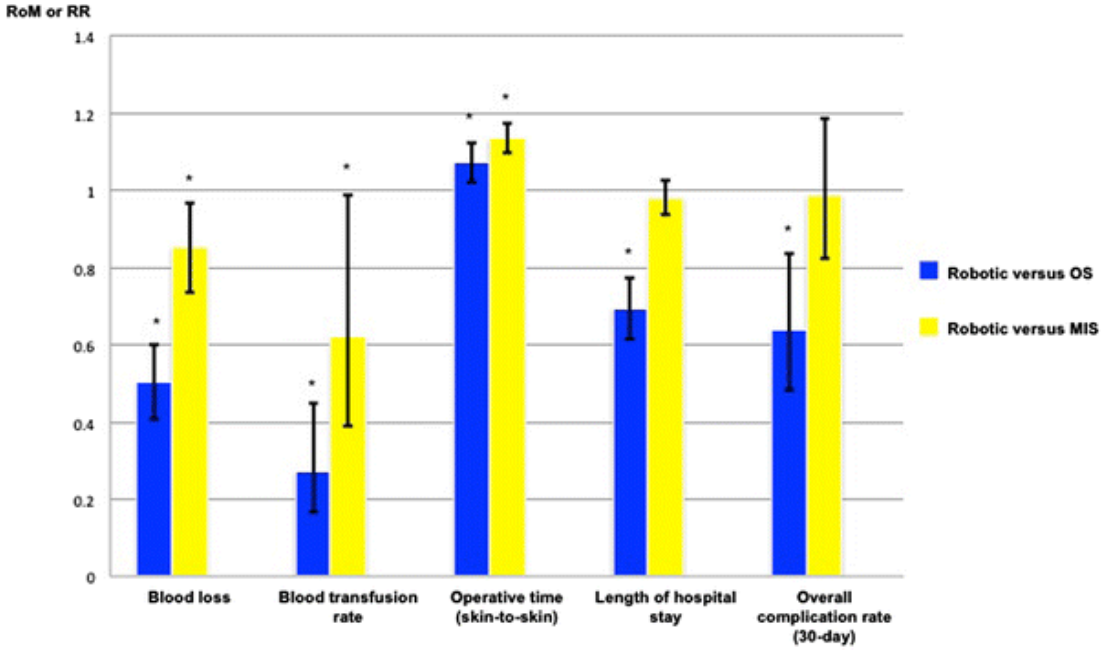


Figure 2.2: Robotic surgery vs. OP and MIS (From Tan et al. (2016))

but similar length of hospital stay (98.2%) and 30-day overall complication rate (98.8%). In both comparisons, robotic surgery prolonged operative time (7.3% longer than open surgery and 13.5% longer than MIS).
Various studies have demonstrated that there still is lack of precise evidence for the advantages of the robotic procedures: this requires the development of new validated and robot-specific methodological tools to assess and evaluate this evolving technology. This includes methodologies to judge element such as specific quality of life (QoL) and patient reported outcome measures (PROMS) combined with robust cost-efficacy and economic analyses.

Endoscopic robots The endoscopic robots are the kind of robot I'm interested for my thesis; they follow the "first generation" of robots, more limited robots that are able to perform single tasks without a lot variability in term of movements and tissues (Ashrafian et al. (2017)).

This "second generation" of surgical robots have introduced the soft-tissue surgical capability: they are able to remove tissue volumes of predefined size. In addition there was a strong request from the surgeons and health's institutions for a system that could allow augmented MIS surgery for laparoscopy using stereo-endoscopic platform. for those reason, this generation have been the greatest expansion of robotic surgery yet.
Camarillo et al. (2004) divides the robot in three categories:

- Passive role: the role of the robot is limited in scope, or its involvement is largely low risk;
- Restricted role: the robot is responsible for more invasive tasks with higher risk, but is still restricted from essential portions of the procedure;
- Active role: The robot is intimately involved in the procedure and carries high responsibility and risk.

While still being limited, those kind of robot are between the Restrictive and Active role definition, they're not autonomous yet but the motion of the instrument doesn't depend just on the surgeon control.
The robots allow the access to tissue in places and organ system that would otherwise be very difficult to reach without recurring to open surgery, without causing torque or sheer to the tissues. Some tasks could be performed by an human with basic MIS tools, but they're difficult to control and they lack force feedback. The use of this kind of Surgical robots can then be viewed as "extending or enhancing human capabilities" rather than replacing humans. (Camarillo et al. (2004)
In 2000, the daVinci robot (Intuitive Surgical Inc, Mountain View, CA, USA) was the second endoscopic robotic system being commercialized; the first one was the Zeus robotic system from Computer Motion, company that has been acquired by Intuitive Surgical in 2003; for this reason, the daVinci become the only commercially available endoscopic robotic system, becoming the most sold in the market.

Different robot platforms share console and various utility similarities: one platform can be used to train surgeons for teleoperative experience with an another

Figure 2.3: The complete daVinci robot system

(Deanna et al. (2014)). As those second generation robot are the most widespread and established platforms in current clinical use, they are the platform on to which many novel technological innovations are currently being applied. These range from improved surgical visibility and visual information transfer to improved robot-survival interactions ranging from haptic tactile feedback to ease of application in surgical environments.

2.1.2 The robot (daVinci Surgical System)

The daVinci Surgical System has been responsible for most of the robotic surgery procedures that are being performed since the 2000's. This system was first used on human subjects in 1997 (used in coronary artery surgery as a proof-of-concept operative principle) and has ultimately become the first FDA approved robotic laparoscopic surgery system in 2000. The daVinci seems particularly favoured in surgery of the pelvis (in urology and gynaecology): over all radical prostatectomies performed in the USA between 2003 and 2010, the national robot-assisted radical prostatectomy (RARP) adoption rate increased from 0.7% to 42% (Murphy et al., 2006).

Its supremacy in the market has being achieved due both to its unique position in the market (being the only one available) and its multiple features that makes it very versatile.
The history of the daVinci is well described in a paper by Camarillo et al. (2004).

Figure 2.4: The daVinci robot from Intuitive Surgical

The console The daVinci offers an ergonomic console and a seat from which the surgeon can operate remotely from the patient; the console offers an in-line eye-hand control of the instrument, together with a 3D stereoscopic image that offers depth perception and zooming capabilities. Some new iterations offers also a laser targeting function.
The computer translates the surgeon's hand movements into the same movements of the instruments, allowing the suppression of physiological micro-motions of the

surgeon's hand that will be not transmitted to the robot's instruments.
Abbou et al. (2017) cites the fact that the integration of preoperative imagery (computed tomographic and magnetic resonance imaging) and intra-operative video to guide the surgeon could be an exiting development for this technology and is interesting to consider the fact that my thesis is framed in a bigger plan to discuss if such thing could actually be feasible.

The arm instruments The current iteration (as of 2017) of the daVinci, offers a 4-arm system: to each arm an different instrument can be mounted. The robotic arms are mounted on a cart: one of the arms holds the high-resolution three-dimensional endoscope while on the others specialised instruments can be mounted.
The "EndoWrist" instrument, a small mechanical joint, allows for 7 Dof (degrees of freedom), the same amount that the human wrist normally enjoys; instead, current laparoscopic instrumentation (as of 2006) allows only 4 DoF. This allows to complete complex micro-surgical tasks and grants easy access to usually "hard-to-reach" areas of the body.
The instrument tips, or end-effectors, are designed from of traditional surgical instruments, to allow surgeons to have a similar tool-tissue interaction as before; they can be sterilized and interchanged during surgery. The instrument will so fulfill various function like a stapler, scissor or haemostatic devices; they can be mounted on the robot depending on the needs of the specific surgery. Other instruments could be a laparoscopic insufflator and the camera control 3D image system described in Abbou et al. (2017).
The surgeon receives some force sensation, from the instruments: This haptic feedback is currently limited to interaction with rigid structures, such as tool-on-tool collisions, and not from the interaction with soft tissues.

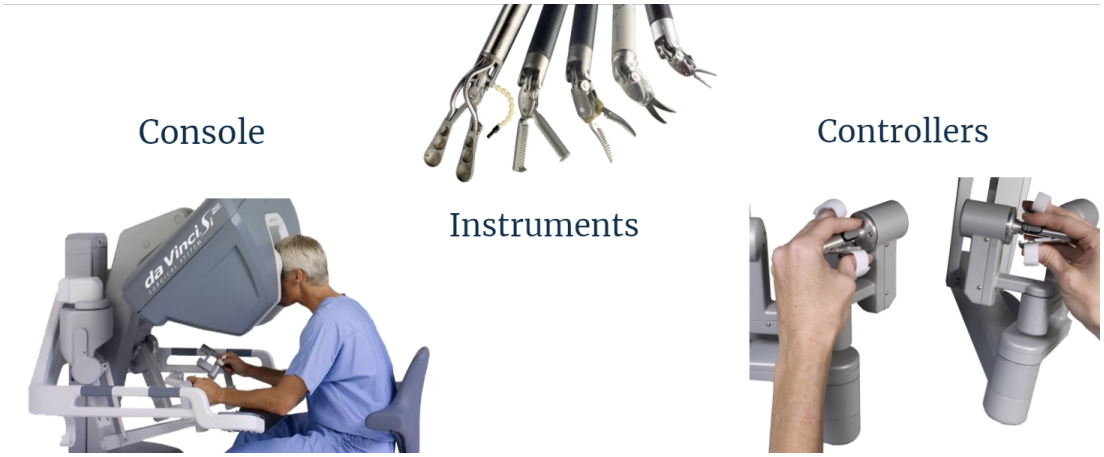


Figure 2.5: Console, controls and instruments of the daVinci robot

2.2 Robot-assisted surgery in Urology

The field of hurology is one of the firsts where robots have demonstrated to be a viable option for performing surgery; in this field, MIS technique are still very used, but the robotic ones have become standard procedures. Many techniques has been developed, especially for cancer treatment of kidney and prostate.
When treating kidney cancer, if the tumor is not too large or problematic, it would be possible to have partial nephectomy instead of total: this would allow to spare the most part of the kidney but it is a very complex surgery which requires precision and dexterity in an hard-to-reach position.
Robot-assisted partial nephrectomy (RAPN) is considered a feasible MIS alternative to open partial nephrectomy (OPN) for the surgical treatment of renal tumors; Casale et al. (2019) suggests that RAPN results optimal outcomes in the majority of individuals despite tumor complexity.

2.2.1 Robot-assisted partial neprectomy (RAPN)

Robotic-assisted partial nephrectomy was first described byGettman et al. (2004) and has been perfected over time since them, increasing popularity. Today, the

RAPN is considered a valid alternative to open partial neprectomy (OPN), being much less invasive, and having comparable results (Vittori (2013)).
Before the development of this new technique, the laparoscopic partial nephrectomy (LPN) was the standard; as stated before, the robotic surgery have some advantages when compared with the traditional techniques, such as enhanced dexterity, greater precision and a 3D magnified view of the surgical field. Those advantages will make RAPN a feasible option for the treatment of renal masses (Long et al. (2012)).
When compared with OPN, the RAPN has lower blood loss and surgical complication rates, with a generally shorter hospital stay; in comparison with LPN, RAPN has with a shorter learning curve and shorter WIT (warm ischemic time), with the advantages of traditional minimally invasive surgery (MIS) techniques (Zhang et al. (2013)).
Having analyzed the reports in the literature, Casale et al. (2019) states that based on their findings, RAPN should be considered as an effective alternative and equally feasible to OPN and LPN when treating renal masses; it is associated with low complication rates and conversion to open surgery rate ensuring optimal oncological outcomes. Additionally, they would support the use of RAPN even in patients with complex renal tumors. Porter and Blau (2020) reports similar findings.
Quraishi et al. (2020) reports on the latest techniques for performing RAPN, giving an example of a procedure using a daVinci robot. RAPN allows the removal and reconstruction of the kidney, performed through multiple little incisions, as opposed to a single open incision in OPN which is much larger and painful. Also LPN may be performed using smaller incisions but it can be technically challenging and more often converted to a total nephrectomy.
Many techniques are being developed to improve RAPN, for instance Porpiglia et al. (2020) are studying a way to overcome ultrasonic guidance, using a 3D model of the organ to use augmented reality during the surgery.

2.3 Limitation to robotic surgery

Cost A contemporary daVinci robotic platform costs approximately €1.8 million, with a yearly service charge of €145000 and instrument cost of approximately €2300 per case. This remains beyond the financing capability of the majority of public hospitals.
While robotic surgery costs remain high compared with open and MIS cases, there is increasing evidence to suggest the long-term cost efficacy of robotic approaches compared with traditional open operations (such as for radical prostatectomies); this is largely as a result of lower overall complications, lower incontinence and lower sexual dysfunction costs that robotic prostatectomy provides (Hughes et al. (2016), Tandogdu et al. (2015), Bolenz et al. (2010)).

Learning curve The surgeon has to be retrained for the new procedure, even if he is already experienced with that kind of operation performed without the use of a robot, using a MIS traditional platform (Porpiglia et al. (2013)). It typically takes 150–250 cases to achieve the learning curve for operative time (Wolanski et al. (2012)); comparing learning curves and proficiency rates between procedures and techniques can be problematic as utilizing open, MIS and robotic procedures to achieve the same end result may not follow the same steps and therefore are difficult to compare clinically and statistically. Additionally, most practising surgeons become familiarized with MIS techniques before they go on to practice robotic surgery, so that a true comparison of their learning curves could be biased.

Environmental and operational limitations Most current robotic platforms needs sufficient theatre space that can accommodate the large dimensions of the devices. In addition, the staff has to be trained specifically for the robot, not just the surgeon but all the staff has to be familiar with the set-up of the robot before surgery, being able to manage all the complexity associated with the robot.

Smaller and less complex devices could in the future address this kind of problems, allowing for an ease of transporting between different surgeon sites or for an easier repairing procedure, resulting in an increase of the accessibility and the general adoption (Ashrafian et al. (2017).

Feedback to the surgeon While the robot has advantages in terms of dexterity, the lack of a proper force feedback with the soft tissues is an important issue: for this topic there is a lot of research currently ongoing but nothing yet in the market (Ortmaier et al. (2007), Bahar et al. (2020), Okamura (2009)).
The last important issue, is related to the camera: during the surgery, the S. Martino's hospital surgeons have noticed that the view from the camera can become quite unclear and the surgeon might lost track of the precise position of the organ. For this reason, an application of **augmented surgery** could be very useful: a computer generated image of the organ, result of a simulation, could be super-imposed onto the real organ on the stream from the camera; the simulated organ should deformate and be displaced in the same way of the real organ.

2.4 The future of robotic surgery

The future of robotic surgery depends different aspects; the first is for sure Technology: the continual research and its application will result in better robots, using cutting edge technologies and know-how from different fields to improve the surgical precision and the range of cases in which can be applied to have better clinical outcomes.
If it will be possible to decrease the cost and increase the cost-efficiency for each individuals, more institution will be able to afford a robot surgical system, increasing the number of evidence to better understand its efficiency and effect.
The introduction of augmented surgery, visualizing the 3D model of the patient on the robot's console, could help the surgeon both before and during the surgery.

2.4.1 Future technological improvements

Considering the future technological advancements, one of the most important fields for the future of the robotic surgery is for sure the one regarding augmented surgery. Many technologies focus on enhance the surgical decision, providing more information to the surgeon to make him more aware of the current situation, using both visual and tactile perception.
Introducing more information would allow a surgeon to overcome operative environmental complexities; for example, can be used to overcome the difficulties of operating on a beating heart, by generating a non-moving "phantom heart" image so as to enable the surgeon to visualise a still heart for performing surgery.

Visualization Technology such as Dynamic View Expansion or Mosaicing have already been introduced in MIS and can offer robotic platforms a wider field of view than cameras (Mountney and Yang (2009), Stoyanov and Yang (2007), Lerotic et al. (2008)). Multimodal visualisation technology is too being applied in robotic procedures, such as augmented reality (overlaying of CT, MRI, ultrasound or other imaging) to guide the surgeon in decisions during the operation; these techniques can be enhanced by improving depth perception, by implementing see-through vision of an embedded virtual object while sustaining the vision of standard operative anatomical landmarks(Lerotic et al. (2007)) or by incorporating in real time changes on the tissue surface. For this application, real-time intra-operative ultrasound (USS) had added a technically simple yet diagnostically powerful imaging modality to robotic surgery and is used extensively for robotic partial nephrectomy; it has potential in procedures where it can help differentiate tissues (Hekman et al. (2018), Mohareri et al. (2015)).
Tissue imaging with photodynamic capture and enhanced microscopy ranging from Narrow Band Imaging, Fluorescence Lifetime Imaging and other techniques, will offer increased real-time visual histological data that can identify tumour cells and margins (Galletly et al. (2008).
Robotic cameras quantify tissue autofluorescence after emitting radiation, this

allow to highlight any microscopic tissue of interest. The combination of this, enhanced diagnostic computation, visual processing tools and machine-learning algorithms (Esteva et al. (2017)) will allow the precise recognition of certain tissues that can be removed with high precision from the robot.
Porpiglia et al. (2020) are studying a way to overcome ultrasonic guidance, using a 3D model of the organ to use augmented reality during the surgery.

Tactile feedback As highlighted before, the lack of tactile feedback is a strong limitation because the sense of touch can be of critical importance in differentiating pathology and making surgical decisions while it is being performed. Increased tactility will offer a new level of tissue perception for robotic surgeons that could be translated into increased precision and safety.
The study of the neurophysiology and trasduction of human tactile perception is allowing the scientist to build wearable haptic systems to offer tactile enhancement: this will allow even smaller operative utensils with higher DoF joint capacity and more advanced kinematics (yun Yao et al. (2005), Pacchierotti et al. (2017), Hammond et al. (2013)).
Still, if the tactile feedback is not perfect, could be more harmful than useful (Overtoom et al. (2019)).

2.5 Augmented surgery

The augmented reality (AR), applied to the robotic surgery, could be a game changer in terms of information to which the surgeon can access while performing the surgery, if necessary.
Reis et al. (2021) have performed a review of the literature regarding the use of augmented surgery and has pointed out that, while actual technologies may have some limitation in term of accuracy, the future is very promising and there is a lot of interest from scientist and researchers to improve them or create new ones.
The augmented reality, or mixed reality (MR), superimposes new objects onto

the view from a camera: to perform such task, many different aspects have to be considered. The visual tracker is a crucial component: it allow the virtual object to mantain the right pose and orientation while the visual from the camera is changing, or to change in the right position if the camera is still.
In robotic surgery, the virtual object might be a 3D model of an organ, for instance a kidney, that has to follow the pose and orientation of the real one while the camera of the robot remains still. The tracker will use visual features or other information to keep continuously track of the real object, a task that could be challenging.
The information from the camera and the tracker can then be fused together to generate a unique video stream: in the case of the daVinci robot, the destination of the video would be the integrated visor that allows for stereoscopic video playback.

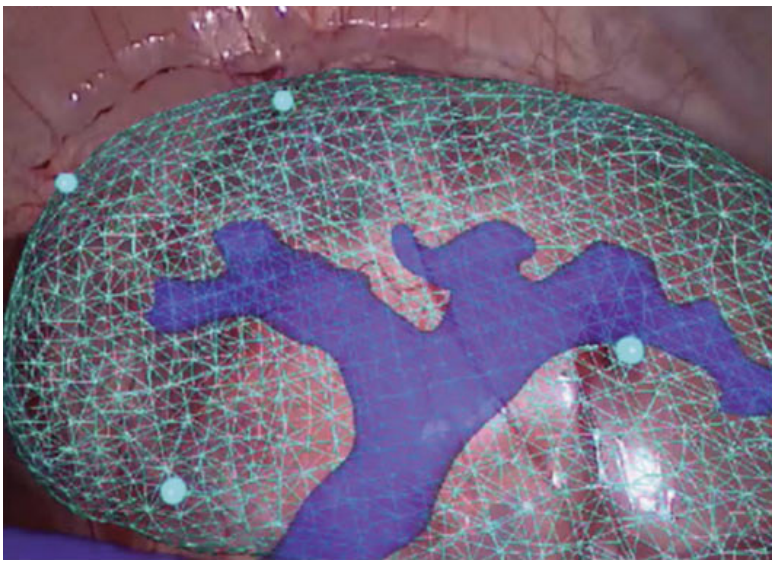


Figure 2.6: Augmented surgery application with a kidney (Kong et al. (2016))

2.5.1 Augmented reality in hurology

It has been demonstrated that the use of augmented reality has a lot of benefits both while educating and while training.

While educating, the students could interact with a simulation, manipulate and interact with it, especially in the 3D perception of anatomy of the organs.
For the training, the main goal are **surgical simulators**: a way for the future surgeon to learn and improve a surgical procedure in a safe and controlled environment. The surgeon will interact with the control instruments and see the simulated result on the visor.
Hung et al. (2015) developed a simulation platform for RAPN that could use both augmented and virtual reality, for a partial or fully simulated environment, which resulted very effective.
At surgery time, MR could provide significant support to the surgeon: both during surgical planning, using information from scan (such as CT, MRI or PET) then during the actual surgery. It would be possible to see the actual organ in 3D, not just in 2D like in standard monitors.
The virtual organ used in the MR environment would be the one of the patient that will undergo surgery: we can define it an application of **personalized medicine**. The 3D reconstruction of organs from the patient's scan have demonstrated very reliable and useful (more on this in Section 3.2.

The main issue lies on the fact that organs are not fixed and rigid objects, they can change in shape and position, especially when under surgery. For the case of the kidney, in addition, in a close-up view the organ look very homogeneous with very few distinguishable visual features, especially with a restricted view; deformable models that are able to take internal organ deformation are currently under development. Artificial markers might be applied on its surface and be used for estimating pose and orientation; in addition there is ongoing research about the possibility of applying inertial sensors on the organ.

Chapter 3

Modeling the deformation behavior of soft tissues

The modeling of soft tissue has become a relevant topic of research, due to both Minimally Invasive Surgery (MIS) insurgence and surgical training simulators. Famaey and Sloten (2008) provides an overview of different models of the continuum-mechanic that keeps in account different properties of the soft tissues. Those kind of models are directly based on the laws of continuum mechanics; they provide high fidelity and are less expensive that other kind of models (like the mass-spring one), using finite element methods and boundary element methods (described in Meier et al. (2005)).

3.1 Material's properties

The models that allows the Computation of realistic surgical tool–tissue interaction forces and organ deformation through the continuum mechanics approach first requires understanding of physical laws governing the behaviour of soft tissues. From Fung (1993), we have a base to define various properties of the soft tissues that can be modeled.

A material is called hyper-elastic when it assumes its original form after removal of the applied forces; a second condition is that the acquired deformation or stress

pattern is independent of the followed path to obtain this deformation or stress pattern (Bonet and Wood (2008)). Describing the Hyper-elasticity (for soft tissues, Gasser et al. (2005)) via the strain energy density function (SEDF), is a common way to describe the material behavior as a relationship between stress and strain (Hibbeler (2005)).

A way to simplify the model, introducing some limitations, is to consider the elasticity as linear, imposing a linear relationship between stress and strain. Misra et al. (2007) noted that linear models are less realistic, also because soft tissue behavior can be considered linear only for small displacement and deformation, which is not the case for most surgical application; also, the linear model is not invariant with respect to rotations.
Cotin et al. (1996) suggests that the linear elasticity can be a reasonable approximation when computing soft-tissues deformation.
To describe the elasticity more accurately we have to use nonlinear models; Picinbono et al. (2001) analyzes the St. Venant–Kirchhoff elasticity and found it realistic.
According to Horgan and Saccomandi (2003), the nonlinear material behaviour can be described using power law models: for instance, one of the most important models for soft tissues is the Fung model (Fung (1967)): starting from that, many other models has been developed.
A viscoelastic material, will not immediately resume its original form after removal of the applied forces, differently from an hyper-elastic material, because it is history-dependent and not path-dependent. Besides a nonlinear stress-strain relationship, Fung (1993) also identified a hysteresis loop in cyclic loading and unloading, stress relaxation at constant strain and preconditioning in repeated cycles.

An other way to simplify the modeling of elasticity, we can introduce the concept of pseudo-elasticity: we treat a preconditioned material as one hyper-elastic material in loading and another elastic material in unloading (Vito and Dixon

(2003), Holzapfel and Weizsäcker (1998)).
In addition, different Soft tissues can be internally composed by different layer with different components, models should be able to take them separately into account and compute the total tissue strain energy as a sum of the components of each layer.

To be suitable for a definitive surgical simulation, a model should be able to incorporate the tissues damages during deformation, micro-structural changes cause a reduction in the mechanical strength (Natali et al. (2005))), changing the hysteresis and other mechanical properties over time. Models such as Fung's pseudo-elastic model are not suitable because they describe the behaviour of tissue in a preconditioned state with a constant hysteresis.

3.1.1 Constitutive Models

BEM Kim et al. (2007) introduces a method multi-resolution model technique that allows physically-based real time simulation. In particular, this method allows to include deformation as response of intervention with surgical instruments. This technique uses a coarse model to represent the whole organ but it is enhanced locally using a mesh subdivision and smoothing algorithm: compute the global deformation would be numerically and computationally expensive, and it is not feasible in the case of a surgical simulation because it requires real-time performances. The tool-tissue interaction is restricted just in the neighborhood of the contact point on the organ's surface. An high-resolution model is required for this region to obtain an accurate simulation, that kind of resolution is not needed for the rest of the organ.
The technique used for the global deformation (the rest of the organ) is based on a Boundary Element Technique (James and Pai (2003)), a discretization of the integral equations of motion with respect to the model surface. Using BEM, the surface of the object is divided into *patches* and the equations can be solved efficiently, using acceleration techniques such as Fast Multi-pole Method (Greengard

and Rokhlin (1997)); this also allows real-time interaction in surgical simulation. For the Local Deformation, instead, they introduced a local subdivision algorithm and a relative smoothing one, starting from a computer graphic algorithm (Vlachos et al. (2001)) that divides the surface in triangles, iteratively smaller, than deformed depending by the applied forces and then interpolated and smoothed to obtain the new deformed surface.
According to Kim et al. (2007) this technique has advantages over the FEM technique (for the modelling and analysis of elasto-static problems (Brebbia et al. (1984)), that i will describe later, but the authors also states that the FEM has advantages regarding the modelling of non-homogeneous and non-linear materials and that their technique could be possibly used with FEM.
The BE and FE methods are constitutive models, they allow to simulate more physically accurate deformations, incorporating real physical material properties. The main difference between them is that the BEM computes the deformation over a surface, the FEM instead computes it over a volume.

FEM From Misra et al. (2008), Chanthasopeephan et al. (2007) and Chanthasopeephan et al. (2004) we have a good description of what FEM is and how can be used to model soft tissues.
The Finite Element Method is a numerical technique that allows to solve, up to a certain error, field equations; in the soft tissue deformation it solves the equations of continuum mechanics. It origins in the 40's (FEM historical review, Pelosi (2007)) and it is first used in the current form in elasticity and vibration analysis (Cook et al. (2007)).
A FE method divide the volume into number of sub-regions called elements, much simpler that the original one, where the adjacent elements are connected via nodes. The result of the simulation will output the nodes displacement, this will characterize the global deformation. Each element has its characteristic elastic behavior, depending by the element's material and geometric properties; we

can than use interpolation functions to approximate the behavior around a certain node.

The soft tissue deformation is particularly challenging because the biological tis-

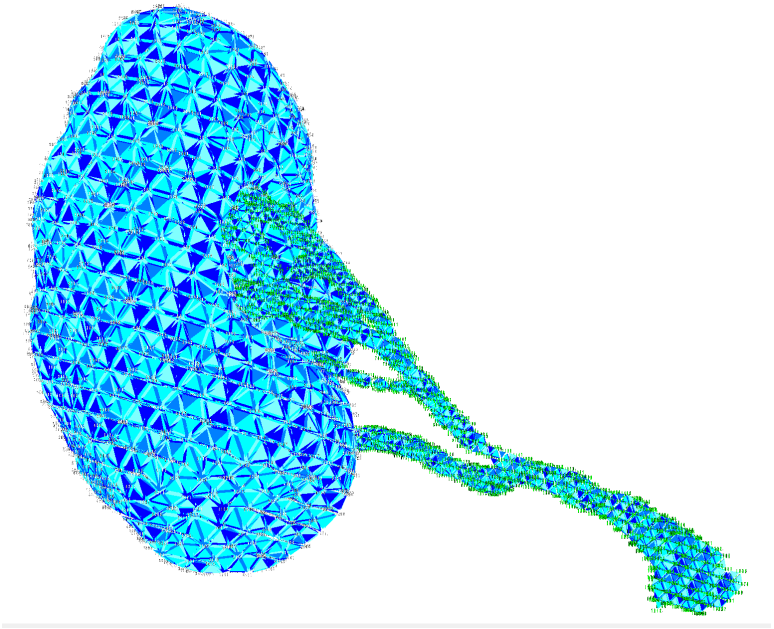


Figure 3.1: Representation of a FEM force field in SOFA

sues are not homogeneous and anisotropic and have nonlinear constitutive laws; as of today, most FE models are optimized for linear elastic problems, allowing a small variation into the material properties, but there are many research ongoing in this field.

The FEM simulation can be very precise, but adding more and more elements to it could make it computationally expensive: a trade-off must be made, especially if the goal is to use that model during a surgery in real-time.

It is also challenging to obtain the material parameters of an organ, together with its precise shape.

The Linear elastic FE models are so the most used to model tissue deformations, they are computationally efficient and easier to setup, considering isotropic and homogeneous materials. To be able to use them in real time (both linear and

nonlinear elastic models), some techniques can be used to simplify the computation (Bro-Nielsen (1998)).
Cotin et al. (1999) have used data from a CT scan to generate a 3D anatomical model of the organ and then used this method to simulate its deformation. This method is effective for small strains(1–2%), but Kerdok et al. (2003) noted the same cannot be said for large strains. Other examples can be found in DiMaio and Salcudean (2005), Picinbono et al. (2000), Wu and Heng (2005), while a linear visco-elastic model can be found in Samur et al. (2007).

The hyper-elastic FE models are a better way to simulate the large deformation that will occur during surgical procedures; it is necessary, as stated before, to identify an appropriate strain energy function that allows to find the constitutive stress-strain relationship. Still, the characterization of nonlinear behavior of real tissues is a complex task, but we can find some successful example in Liu et al. (2004) and Wu et al. (2001).

An even more realistic representation is a Visco-Hyperelastic Finite Element Models, that combines both properties, Puso and Weiss (1998) were the first to implement this kind of model for soft tissues.

Some methods to approximate the FE models exists, for instance the Active Contour models, technique that comes from computer graphics (Kass et al. (1988)).

3.1.2 Heuristic (or particle) based models

A different kind of modeling is the heuristic-based one: those models represent the geometry as deformable splines, for instance the mass-spring-damper model, linked volumes or the mass-tensor model. While their physical realism is limited, they are frequently used in cases when the simulation has limited time constraints, due to their computational simplicity.

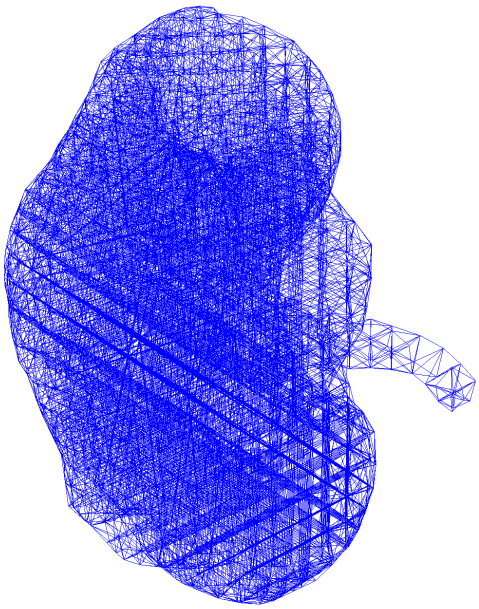


Figure 3.2: Representation of a MSM in SOFA

MSM Firstly introduced by Hrennikoff (1941), the mass-spring model is an example of linear model, Cotin et al. (1999) shows a general analysis of the method and a possible implementation in soft tissues simulation in a surgery simulation environment, including a patient-based 3D organ reconstruction, procedure described in Montagnat and Delingette (1997). This approach includes some pre-simulation computations performed using FEM.
In general, the mass-spring model consists in a certain number of particles (composing the mesh of the object) connected by a network of springs (and dampener, in the case of the mass-spring-dampener); each particle movement will influence the position of the other one connected to it via a spring.
Villard et al. (2008) proposes a new framework, based on mass-spring model used in combination with the tensional integrity method. This allows to have a deformation on the organ that is more consistent that the sole mass-spring model, especially in retaining the shape of the object under large forces without degradation, allowing it to handle larger deformations.
Xu et al. (2018) proposes a new way that allows to simulate with this method

not only the elastic properties but also viscoelasticity and non-elasticity.

3.1.3 Hybrid approaches

We also have some hybrid approaches, that uses different methods to simulate different aspects of the material or to improve the computational time of the simulation.
Hu and Desai (2004) describes a hybrid visco-elastic model, matching up their experimental results with a pig liver. Their model uses linear and quadratic expressions to relate the measured force-displacement values, which are valid for both small strains (up to 16% compression) and large strains (around 16-50% compression).
Zhu et al. (2008) proposes and approach based on BEM, for precise global deformation and the use of meshless shape matching method to achieve low latency; this method is addressed to interactive models, that should give results in real time so speed an low latency are paramount problems. The meshless shape matching is an iterative approach that can be very fast, used to simulate the vibration behavior of the tissue. A state machine will switch between the two methods, analyzing the force on the object; the first method will be used to compute the deformation, responding to external forces, when those are removed from the surface, the second method will be used to restore the original shape.

3.1.4 Extract material paramteres

For some models, it is possible to derive a formula to directly calculate the parameters. For instance, Gelder (1998) was able to compute the parameters for a Mass-Spring model in a static state, where the materials are non-uniform but isotropic.
An isotropic material is a material whose local deformation in response to force is independent of the direction in which the force is applied. However, in a non-uniform material the response varies with the position where the force is applied.

For 3D tetrahedral meshes, some parameters can be directly computed for formulas, however, often it is not possible to do such calculations for models that rely on complex constitutive material laws as in the FEM.

Material parameters based on bio-mechanical soft tissue behaviour
The major advantage of using a differential/integral equation-based technique for deformation modelling is the ability to use a material model. When using a model, is possible to assume linear isotropic elasticity: this implies that only two independent parameters, Young's modulus and Poisson's ratio, are necessary to simulate the response of soft tissues.
The **Young's modulus** (or elastic modulus) is a mechanical property that measures the tensile stiffness of a solid material (model the stress-strain relationship of a material) while the **Poisson's ratio** is a measure of the tendency of a material to get thinner in other two directions when being stretched in one direction (measuring the compressibility of the material).
Those kind of parameters can be measured on tissues, both ex-vivo or in-vivo, using devices such as such as TeMPest, Ottensmeyer and Salisbury (2001). To measure those parameters, some known forces are applied to the tissue and the displacement of some known point in the volume is measured.
Most of the techniques developed in the past are ex-vivo, in Fung (1993) is possible to find one of the first comprehensive study on various human tissues.
There are also some examples where the mechanical properties of human or animal tissues are directly extracted in-vivo (Tay et al. (2002), Luboz et al. (2012), Samur et al. (2007), Hollenstein et al. (2009), Schwenninger et al. (2011), DiMaio and Salcudean (2002)), some of them also contains a test comparing the real tissues with the simulated one using the obtained parameters.
For the kidney and the blood vessels, study reporting the parameters that can be used in a simulation can be found in Karimi and Shojaei (2017) for the kidney and in Li (2018) for the blood vessels.

3.2 Physical models

3.2.1 Obtain physical model of an organ from a patient scan

In 1994, a full set of 3D human anatomies was published (Ackerman (1999)); from that moment, various improvement has been done in the field of 3D reconstruction from traditional medical imaging techniques.
An important step in surgery simulation would be to become able to generate a personalized 3D model of a patient organ, from a medical scan. Various example and methods can be found in Dawant et al. (2001), Pan and Dawant (2001), Andrew (2000), Lorensen and Cline (1987), Montagnat and Delingette (1997).
Cotin et al. (1999) is one of the earliest work and analyzes all the step that leads to the simulation, from the CT scan to the generation of the geometric modeling and then obtain the physical model from biomechanical experiments, ending up with the final model that can be used for the simulation.
A comprehensive and extended example of a pipeline for this kind of procedure is well described in Crouch et al. (2007), Chacko and Sawant (2011) and Lee et al. (2018); this would allow to automate the setup of a FEM simulation, to be than used for a surgical simulation. The input data in this case comes from a patient's CT scan.

3D reconstructed organs Clements et al. (2011) used the method proposed by Dawant et al. (2001) on multiple patients and founds that the obtained error was statistically acceptable and usable in a real scenario.
About the accuracy and the usefulness in general of those geometrical models, many studies have reported that the surgeons have found them very useful in the surgery preparation (Favorito (2018), Schiavina et al. (2019), Yaxley et al. (2016), Wake et al. (2017), Shirk et al. (2019)) and have resulted very close to the real organs (Michiels et al. (2019),Porpiglia et al. (2018), Porpiglia et al. (2019)).

3.2.2 3D Model building pipeline

In this section i will present the pipeline that i used to build the mesh model that can be imported into the SOFA simulation as input for the FEM method.

The data about the shape of the organ comes from a CT scan of the patient;

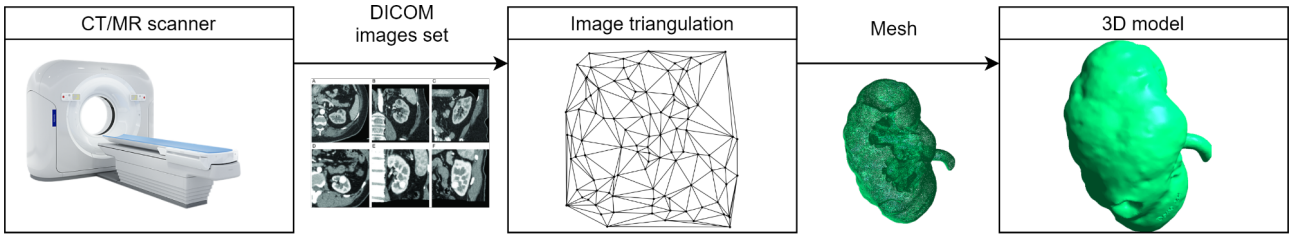


Figure 3.3: 3D model building pipeline

the CT scan produces a dataset of 2D sliced radiological images. The resultant DICOM database has been transformed into a surface 3D model (in the ".stl" format) using the software *Materialise Mimics inPrint* by a specialized technician. The scan has been performed by using a contrast agent: this allow to segment independently different anatomical parts and generate a model composed by multiple parts. In this case, three parts are segmented: the kidney and the blood vessels, the kidney and arteries.

A surface 3D model is not enough to simulate the organ: a volumetric mesh is necessary. I made some initial modification to the 3D model using *Meshmixer* from Autodesk, removing some appendices that were not useful for my purposes, then re-sampled the model mesh using MeshLab, reducing it resolution (being the model way more accurate than necessary), to make the simulation deal with a less defined model and reducing its computational complexity. The volumetric mesh was then generated using *QTetraMesher* (suggested in the SOFA website), and the mesh is modified again using *gmsh* for some final polishing.

The original model of the scan included the organ, a kidney, plus all the blood vessels: the veins, including a small portion of the Inferior vena cava and the arteries, including a portion of the abdominal aorta. The two big vessels, inferior

Vena Cava and Aorta, have been removed; then, the kidney, the arteries and the veins have been separated into three different 3D models.
In the following figures, is possible to see the 3D model of the kidney in addition to the blood vessels and then the separated veins and arteries.

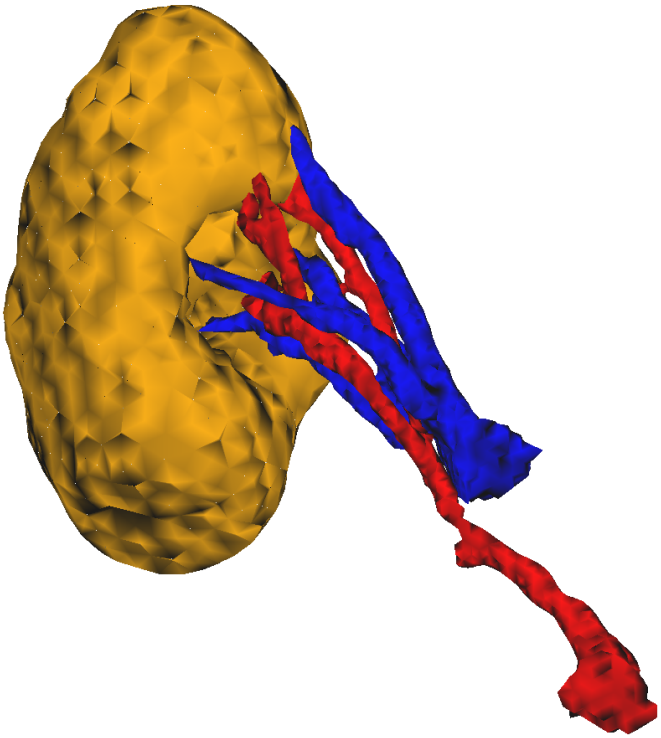


Figure 3.4: 3D model of the kidney and blood vessels

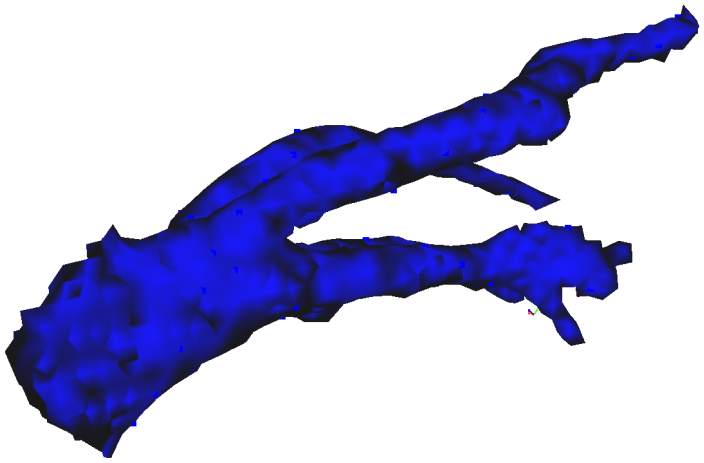


Figure 3.5: 3D model of the veins

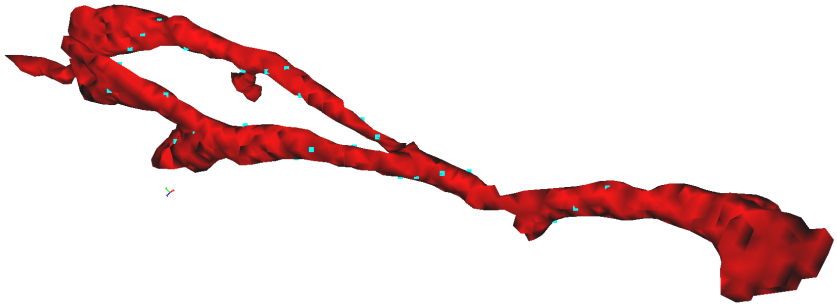


Figure 3.6: 3D model of the arteries

Chapter 4

Simulation and simulation platform

4.1 Simulation platform

The simulation platform that will be used for my thesis called SOFA (Simulation Open Framework Architecture): this platform has been introduced and described in 2007 (Allard et al., 2007), with the goal of performing a multi-model representation: the final Behavior model is composed by sub-models connected together with a sub-mechanism called *mapping*: each sub-model is responsible of representing a particular component (for instance collision detection, visualization, haptic). This decomposition into a set of basic components allows for a clear separation between the different aspects of the simulation and for a reducing of the DoF of each models, making them more easier to compute.
This separation in components is crucial also when considering different objects in the same simulation: each component of each object is computed separately and unless they come in contact and perform in collision, they remain independent.
The simulation in SOFA is graph-based: the scene has a base node and from that it is expanded in child nodes. From the root node, for instance, we can grow a new node that is representing one of the objects that we want to simulate; from

that, new child nodes will represent the various models (visual, collision or mechanical model) of the object.
The article Faure et al. (2012) contains a more in-depth analysis of the SOFA framework; it also contain some example of implementations, for instance an example of an human organ (the heart) and its behavior or a knee-joint mechanics. Comas et al. (2008) points out how SOFA can be efficient in computing nonlinear FEM in the soft-tissues modelling, especially when using the GPU acceleration module.
The geometrical model that can be obtained from the patient's scan, for instance, is in form of a **mesh**, a collection of vertices connected via edges, forming a graph. The faces are usually defined regular shapes, triangles, quadrilaterals or exagons for surfaces and tetrahedrons for volumes; those shapes allows a smooth representation of the deformations (Delingette (1999), Montagnat et al. (2001)).

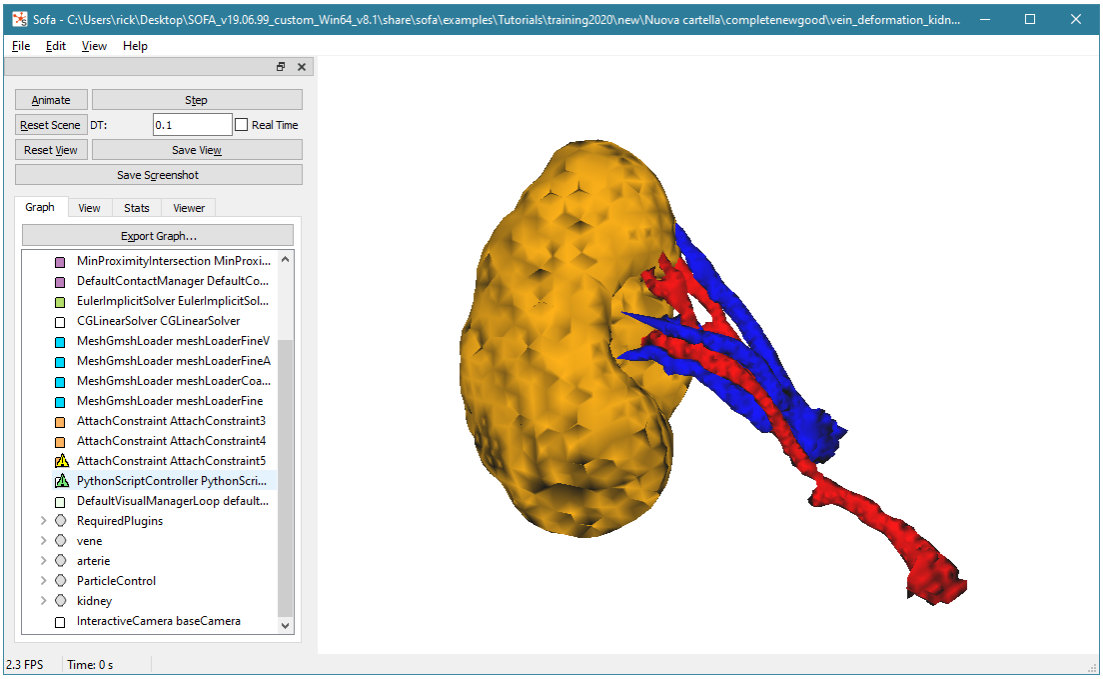


Figure 4.1: User interface of SOFA

4.2 Components in SOFA

4.2.1 Models in SOFA

Mechanical Model This model represents the internal mechanical behavior of the object, can be computed using various techniques, from FEM to a mass-spring type model. The Mechanical (or Physical) model has usually a tetrahedral topology (set using the component $TethrahedronSetTopologyContainer$).
The state vector associated to the model is the $MechanicalState$: this will contain the and the velocity, acceleration and force of each node of the mesh composing the simulated body (the total Degrees of Freedom, DoF, of the model).

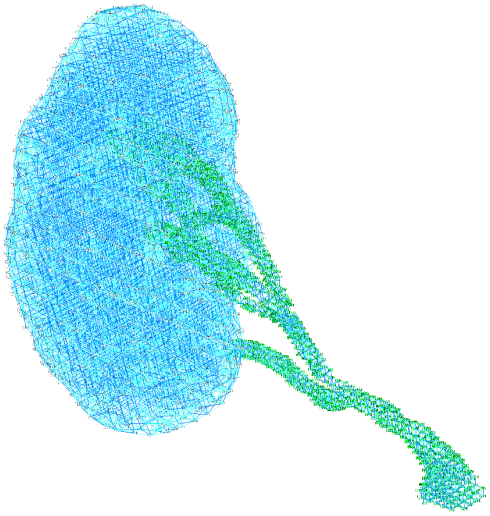


Figure 4.2: FEM model with a tethrahedral shape

As for the DoF of the model, two main templates are present:

- Vec3d: the object has 3 DoF for each node and is used to simulate the mechanics of a body, $[x, y, z]$;

- Rigid: the object has 7 DoF per node and is used to simulate rigid bodies, $[x, y, z, q_x, q_y, q_z, q_w]$.

Collision Model This model is necessary to define the behavior of an object colliding with another one; is important to choose the proper level of precision in the contact handling because the computation process could be quite time expensive. Its consists in a bounding box (or a set of boxes with different sizes, in case of a complex shape) around the physical object; when two bounding boxes of different objects will intersects a collision will happen.
SOFA offers different primitive shapes (for instance sphere or cube) that can be used as simple collision model (even more combined), in addition is possible to use a model from the memory.

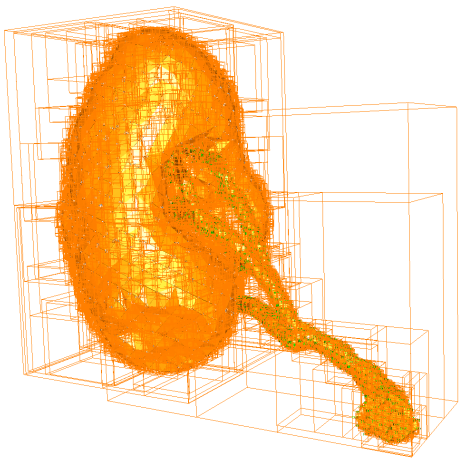


Figure 4.3: Collision model and its related bounding boxes

Visual Model This model allows to attach a visual model to the simulated object, useful for the user to properly understand what is going on during the

simulation; it usually has a triangular mesh with high resolution, to be easily mapped with the Mechanical tetrahedral one.

4.3 Other components

Mapping Is a crucial component of SOFA: it allows to link the models and make them coherent and consistent in changes (usually the mechanical model is the one that enforce the change to the other ones). This allows to use different kind of mesh topology between the various models. For instance, we may use a more coarse mesh in the mechanical model and a finer one in the visual model: we will obtain a good looking simulation that is not too much heavy in the computation of the mechanical behavior, limiting the amount of Degrees of Freedom of the mechanical model.
The mapping defines a function that maps kinematically the position of the mechanical model (the parent) to the collision model or the visual model (the child). The coordinates could be barycentric (for deformable bodies) or local (for rigid bodies); once the correspondence is computed, vectors (for instance forces) could be projected from one representation to another. Is possible to propagate positions, velocities, displacements and acceleration both top-down, from the parent node to the child, and bottom-up, from the child to the parent by modifying a parameter.

4.4 Simulation implementations examples

In this section i will present some examples of simulation regarding human organs and the soft tissues they are composed from which i took some inspiration while building my simulation. L. da Frota Moreira et al. (2013) compares the results from a simulation using SOFA with the experimental ones from an actual needle insertion into an human prostate, resulting accurate and with a low computa-

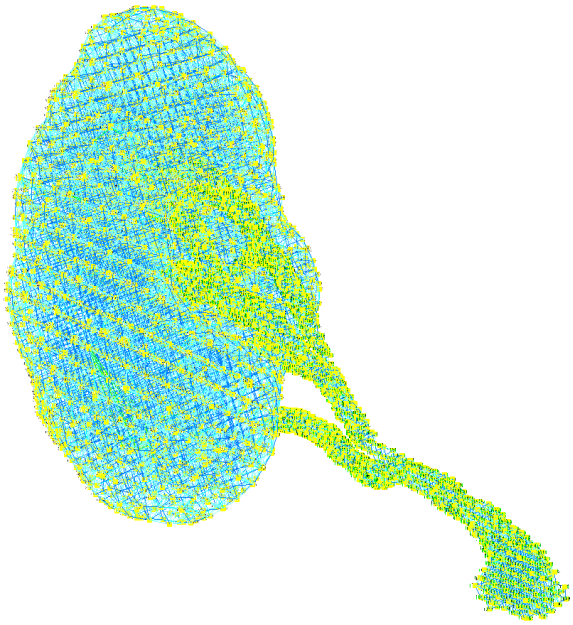


Figure 4.4: Mapping points (in yellow)

tional time, making it suitable even for a real-time surgical simulation.
In Marchal et al. (2006) we can find one of the first example of simulation of this kind of procedure and gives an example of the simulation setup pipeline and the kind of results to expect in terms of displacement.
Horn et al. (2016) describes the steps to simulate an organ starting from the CT scan of a patient.
Jing et al. (2021) have realized a simulation of a kidney, using SOFA, with the scope of build an augmented reality application studying how a kidney deforms when forces are applied to it.

4.5 Anatomy of a basic simulation

The Figure 4.5 describes an example of a simulation graph; the four components are necessary component for SOFA in order to initialize the simulation. The simulation scene is composed of nodes, starting from the node *root* that is the parent node and is the first to initialize.

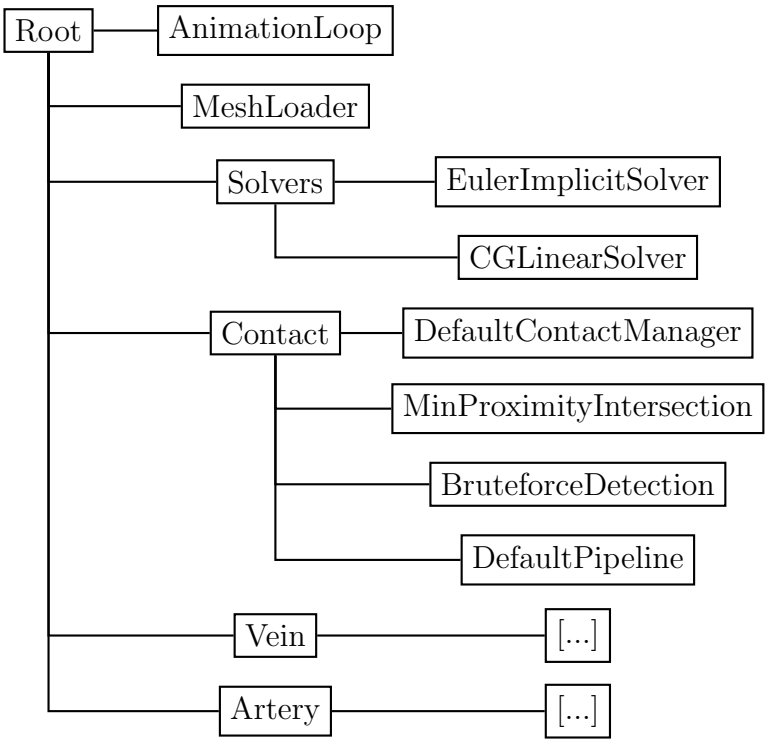


Figure 4.5: Example of a graph defining the scene of a basic simulation

The *AnimationLoop* node is responsible to order all the steps of the simulation and call the solvers when necessary; as first, it calls the the CollisionDetector and computes the collision, if present, then computes the physic of the system, taking in account the defined constraints, calls the solver to solve the resulting linear system and updates the system with the new computed values from the solvers.

Loading the mesh file The *MeshLoader* node (*MeshGmshLoader* in this case) is necessary to load the *.msh* that has been generated from the patient's

scan of the organ; the loaded model can then be used as a mechanical, collision or visual model; if more than one mesh is necessary to add one loader for each desired mesh.

The physical Solvers The *Solver* node contains the desired solvers that are used to compute the temporal evolution of the system using small time steps. The *EulerImplicitSolver* is an integration scheme and uses methods called "implicit": this means that the new time step is computed based on information we already know about it, they're slower than the "explicit" ones (that computes the new step based on the current one, SOFA supports those too) but way more stable.
This first solver describes how the linear matrix system is built, now it has to be solved: the *CGLinearSolver* is an iterative solver, the result is so computed through multiple computations; this method is preferred in bigger system at the direct solver, which computes the result directly. The result will be stored as the status of a MechanicalModel.

Contact Node The *Contact* node includes the sub-nodes that have to detect when a collision between two object happens and have to compute the resultant forces. Depending on the component that are used, the collision can be handled in different ways; *DefaultPipeline* defines how the pipeline of the collision handling is structured: reset of the collision, collision detection and collision response. The reset of the collision clears the data computed from the previous time step.
The collision detection is a delicate state: each CollisionModel is surrounded by a bounding box: if two or more bounding boxes intersects that means that a collision is happening. Here a proximity method is used, the node *MinProximityIntersection* detects when two nodes are getting close, before than the collision actually happens. In parallel to that, the node *BruteforceDetection* checks if some bounding boxes is actually colliding with another (Broad phase). After this, the *DefaultContactManager* (Narrow phase) will detect the model that are in collision and will compute the contact intersection and resulting forces using the penalty method.

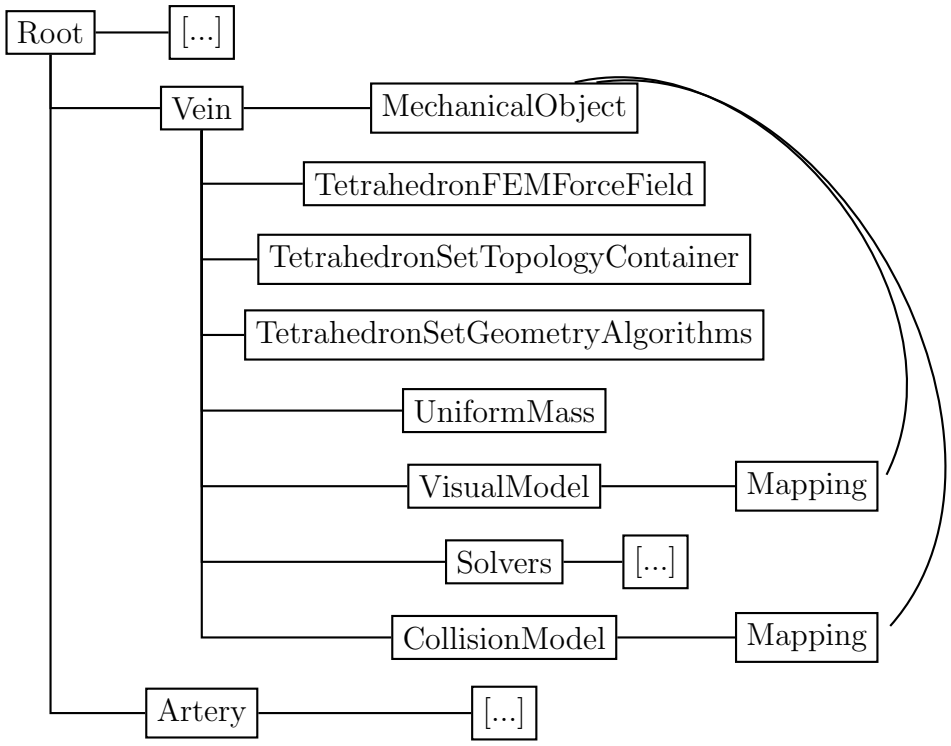


Figure 4.6: Example of an object in a simulation

Simulating object in SOFA The $Vein$ node represents the first object in the scene (we can see in Figure 4.6 its content): it has associated a $MechanicalObject$, a $VisualModel$ and a $CollisionModel$, the models described in the Section 4.2.1.

Mechanical Object and topologies The $MechanicalObject$ in this case is a tetrahedrical model so it is necessary to use the components *TethraedronSetTopologyContainer* and *TetrahedronSetGeometryAlgorithms* to set the topology as tetrahedrical and the relative geometric algorithm that will be used; this last component will include all geometrical function and visualization options for each specific topological element.

Setup FEM simulation A crucial component for the simulation of the internal mechanics of an object is $TetrahedronFEMForceField$: this component is linked with the topology and the mechanical state of the object. The parameter of this component are the *Poisson's ratio* and the *Young Modulus*, described in the Section 3.1.4: those are the material parameters, by using a FEM model (Section 3.1.1), which is a material model, is possible to simulate the material just by defining those two parameters. This component will compute at each time instant the force applied to each point on the mesh, depending on the parameters each point will affect the others and their position, velocity and force. The node $UniformMass$ us used to assign a mass to the FEM model and it will be used when computing forces; it requires a MechanicalObject to store the DoF associated to the nodes, together with the Solvers.
The node *Solver* contains the same solvers as before, this time are associated to the specific object and its DoF and it is not global; they're required to compute how the physics of the object changes over time. In the Figure 4.7 we have a more focused view inside the $CollisionModel$ and the $VisualModel$ of the $Vein$ node; both models are mapped to the $MechanicalObject$ that is part of $Vein$ to keep the model consistent with one another.

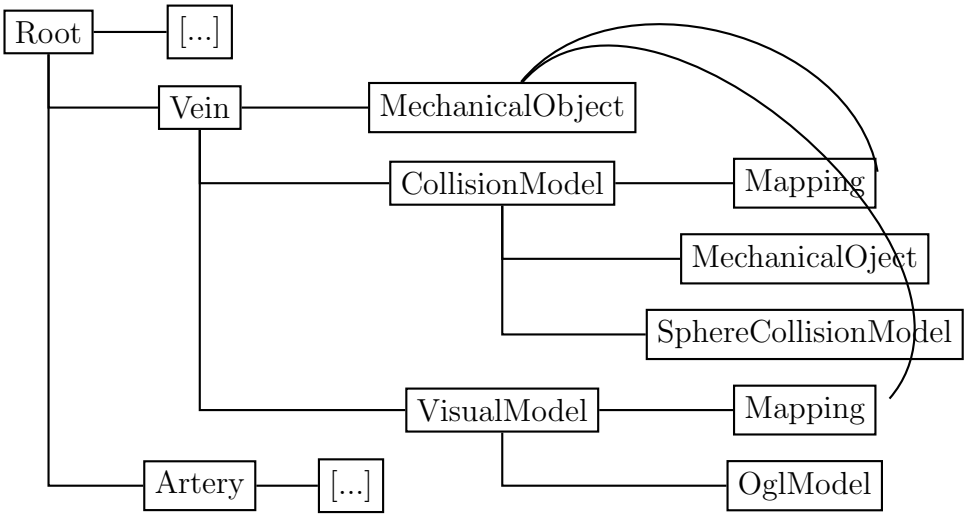


Figure 4.7: Example of an object in a simulation (focus on visual and collision model)

Visual Model As described in Section 4.2.1, the $VisualModel$ is the node that will be used for the visual representation of the simulation, to allow the user to better understand what is going on just by looking at it. The $OglModel$ will use a mesh that has to be previously loaded with a $MeshLoader$ and has various option for the visualization such as the definition of the color of the model, the texture or a shader. The mapping between this and the Mechanical model, will make the Visual model changing shape whenever the Mechanical model is deformed.

Collision model The $CollisionModel$ node defines how the object will react during a collision: the $MechanicalObject$ is an additional mechanical model. This model will be used to store the kind of forces that are generated during a collision; being mapped to the primary Mechanical model of the object, at each time step the Animation manager will take the forces generated during the collision and will apply them to that model. It is then necessary to define which kind of shape the Collision Model actually has: in this case, the collision is computed on a volume that is composed of multiple spheres ($SphereCollisionModel$), as many spheres as the point in the mesh, each one of a fixed size.

At the same level of the *Vein* node we can find the *Artery* node: this node could represent another object, structured in the same way of the *Vein* one, but could also be completely different, using a different force simulation than FEM, using another collision type or other internal algorithms, making SOFA very versatile.

Chapter 5

Setting up the Simulation

While going under surgery, the organ itself, the connected blood vessel and neighbor organs are subjected to various motion and forces. Among all the possibilities, is necessary to choose the behavior that most suits our necessities in term of generating data that can be then used to train a model.
The first approach that i took was to consider the organ itself as deformable (using FEM) and to apply it some forces using a sphere, which should simulate the tip of a daVinci robot instrument. In this way, the simulation would compute the deformation on the organ itself (a kidney) when being manipulated. This approach has presented some limitation: as first, the control of the forces applied to the sphere introduces a lot of complexity, solved by considering forces directed on just one of the three 3D axes; this has made the control of the forces applied to the ball much easier and not completely unrealistic, considering the fact that a force of that kind could be applied by the robot, during a surgery tho that kind of forces are unlikely to be generated.
An other limitation, that has raised after analyzing the behavior of the simulation is that, like in the reality, the simulated organ wouldn't deformate that much, and the study of a local deformation wouldn't be much relevant when considering the pose of the entire organ.
After some discussion with Professor Paolo Traverso of the S. Martino hospital in Genoa, we decided to radically change the scope of the simulation; the goal of

the simulation has than changed from the computation of the local deformation of the kidney to the computation of the local deformation of the blood vessels that are connected to the kidney, while considering this last one as a rigid body for an easier control of position and rotation; the kidney considered as rigid with respect to arteries and veins because it is much more stiff than the blood vessels. For this reason, this kind of approximation the result of the simulation should remain mostly close to the real behavior of the deformation.

5.1 Simulating the blood vessels' deformation

In this simulation three main object are present: $Vein$, $Artery$ and $Kidney$. The main nodes necessary to the simulation are described in the Section 4.5 (Anatomy of a basic simulation).

For the $Vein$ node, is possible to reference the Figure 4.6: some nodes has been added, as visible in Figure 5.1. The $MechanicalObject$ of this node is a mesh that have been loaded using $MeshLoader$. The mesh itself comes from the procedure described in Section 3.2.2: in this case, the mesh will be of the veins alone because all of the three components (the kidney, the veins and the arteries) will be simulated separately because the material parameters are different and so the reaction when subjected to forces will be different.
As described in the Section 4.5, SOFA allow for the use of a material model, using the Finite Element technique (FEM, Section 3.1.1). To perform such simulation, is necessary to know the material parameters (Young Modulus and Poisson's Ratio, Section 3.1.4) of the soft tissue: I've researched the literature and found suitable parameters in Li (2018).
Having setup the simulation using a geometrical model of the human veins and the FEM simulation with the latter's soft-tissues parameters we should obtain as result a realistic deformation of the object while being subjected to forces.
The nodes necessary to setup the FEM simulation (*TetrahedronFEMForceField*, *TetrahedronSetTopologyContainer*, *TetrahedronSetGeometryAlgorithms*) have been

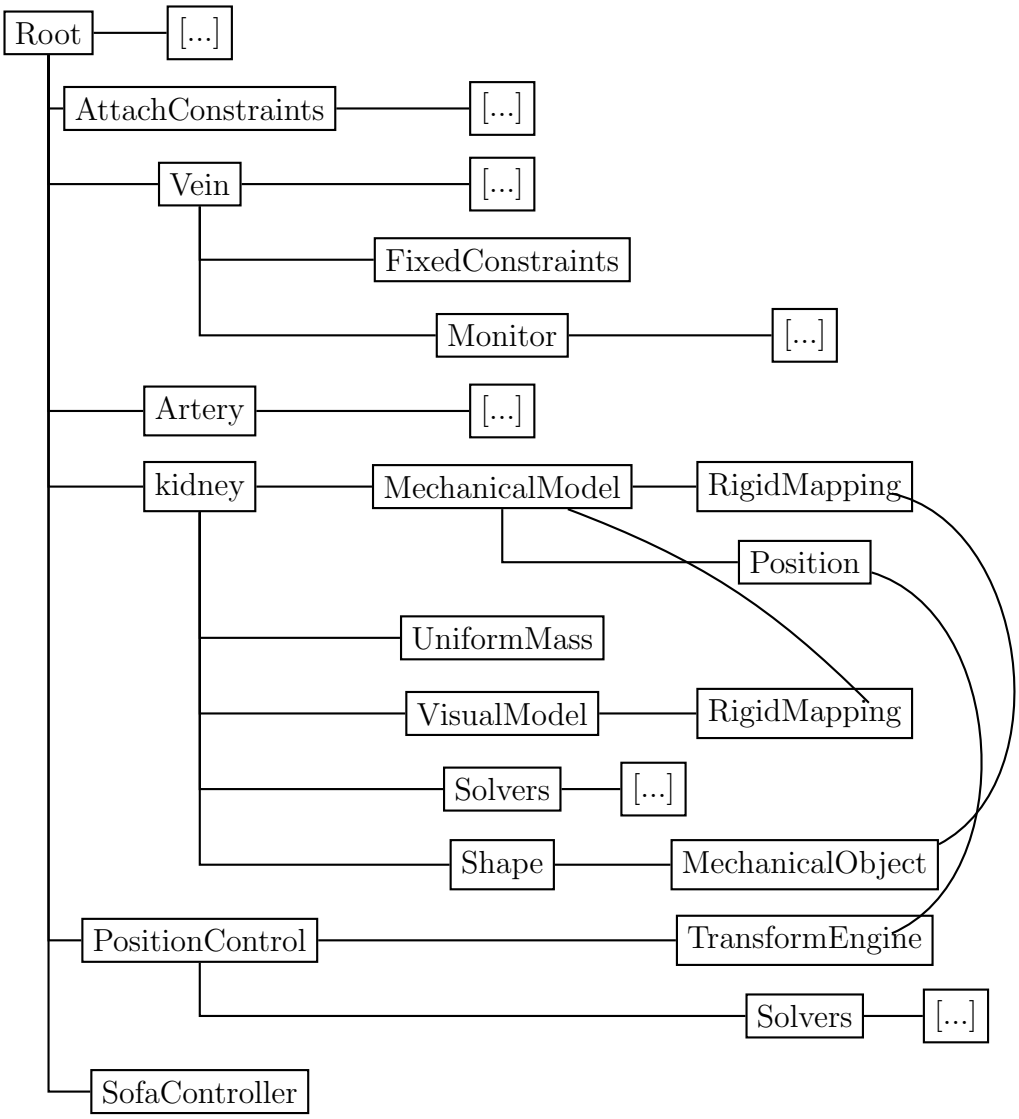


Figure 5.1: Graph of the blood vessels' simulation

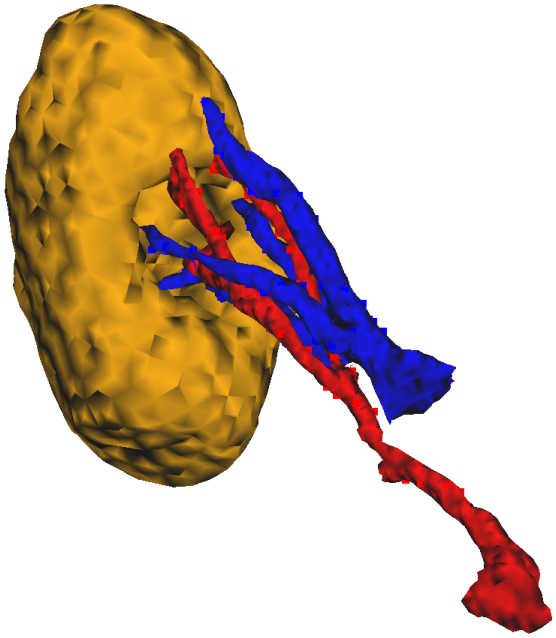


Figure 5.2: Blood vessel's simulation

described in Section 4.5, the same for the Collision and Visual node.
The node *FixedConstraint* is necessary to apply a constraint to some points of the mesh of the object: their position will be fixed as the initial one and their velocity will remain constant and equal to zero during the simulation. The application of those constraints is necessary because some point of the mesh has to remain still: in the real kidney, the veins are attached to the inferior Vena Cava which doesn't deform much during surgery. For this reason, I considered the point that would have been attached to the larger blood vessel as fixed. The points are visible in Figure 5.4, the pink points between the vessels and the world.

The *Monitor* node allow to keep track of the position, velocity and force of some points of the mesh: this will be useful at a future time when analyzing the behavior of the simulation. SOFA allows to draw on the simulation window the trajectory of the points that are being monitored: in this way it can be evaluated

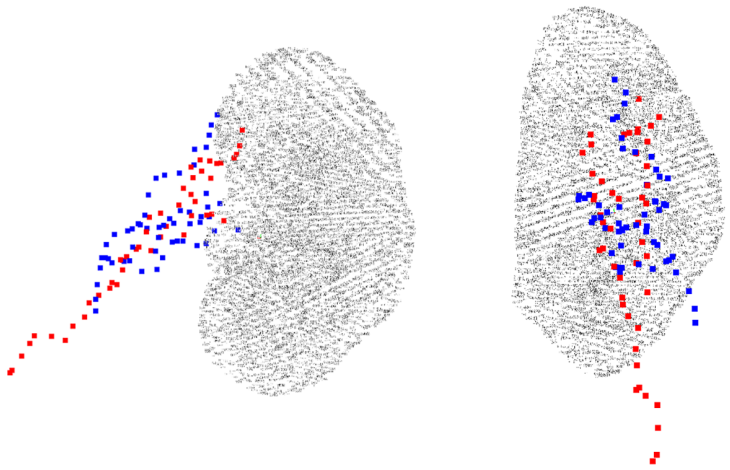


Figure 5.3: The poins that are being monitored, in helical shape

while the simulation is running. The points have been chosen in an "helical" shape, along the lenght of the vessel, from the Vena Cava to the points attached to the kidney. The state of the points will be logged on a file that can be used later (described in section 6.1, "The software architecture").
The *Artery* node is built in a very similar way of the $Vein$ one; it will too being simulated using a FEM technique (the parameters are found in Li (2018) too). It has similar constraints and the points to monitor are again chosen in a "helical" shape.
The $Kidney$ node is built differently: in this case the $MechanicalModel$ represented by a simple rigid particle; its position depends by the $TransformEngine$ that will be described later. This is a simplification but in this case we are not particularly interested to the deformation of the kidney itself (which is much rigid with respect to the veins and artery).
The *Shape* node is used to give the rigid point an actual shape, that itself will be rigid and follow the point pose and orientation; the $MechanicalOject$ inside *Shape* will be a mesh representing the entire kidney but being rigid it Will not deform. This Mechanical model is also necessary to give the Vein and Artery object some points to be attached to.

The $SofaController$ node is associated to a Python script: it will contain the server to which the $ExternalScript$ will communicate to, being so able to give commands. The $VisualModel$ node uses the same mesh of the $Shape$ one, both will be mapped to the pose and orientation of the main $MechanicalModel$.
Two additional main nodes are present: $AttachConstraints$ and $PositionControl$. The first, is necessary to attach the models: some points have been handpicked from the mesh of the vein and arteries to be attached to some other ones in the kidney mesh. This will cause the point to be permanently attached and as result the bodies will be linked during the simulation. The points are visible in Figure 5.4, the pink points between the vessels and the kidney.

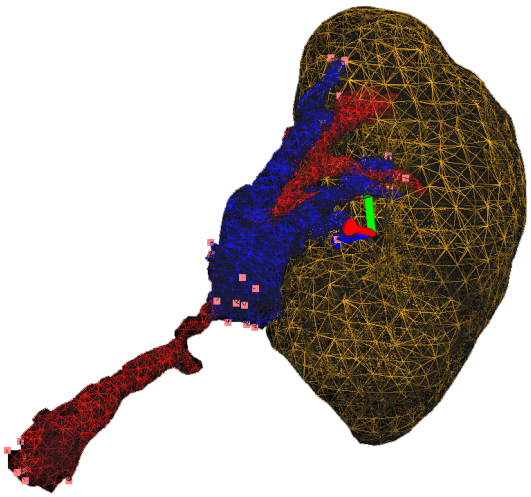


Figure 5.4: Applied constraints (in pink)

The second node allows for the position and rotation control of the $TransformEngine$: its position and rotation will then be used by the kidney's mechanical model as input for his pose and orientation. By giving inputs to the engine, it will be possible to change position and rotation as wanted and those changes will be

reflected in the entire *Kidney* node, being all the models linked together.

The result of the described work is a simulation where a rigid body, the model of a kidney, can be controlled as necessary in position and orientation; two additional bodies, the blood vessels, are attached to some point in its mesh and rigidly attached to some other points in the space. When the kidney will rotate or translate, the blood vessels will follow it (due to the applied constraints) and will deform depending on the kidney's movements. As can be seen in Figure 5.5, the kidney's model has been rotated of around 90° on the x-axis: as a result both veins and arteries have deformed accordingly to the rotation.

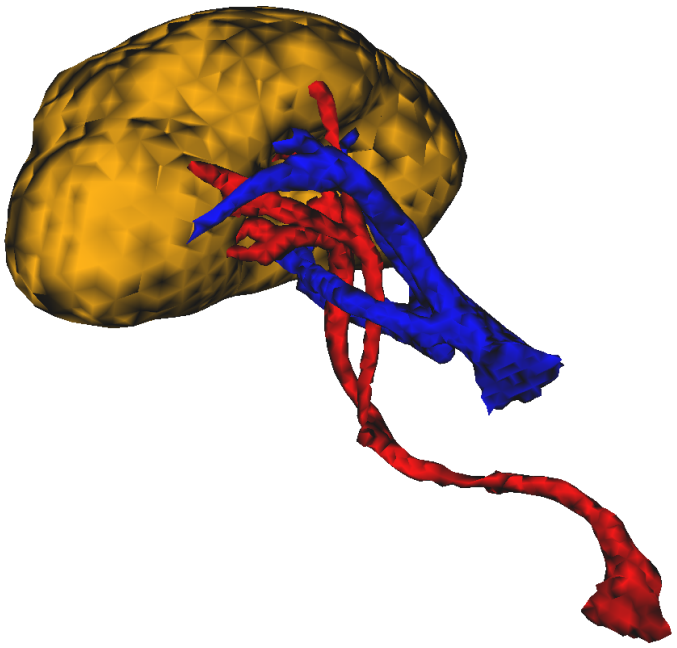


Figure 5.5: The kidney has been rotated of around 90° on the x axis

5.2 Simulating the kidney's Local deformation

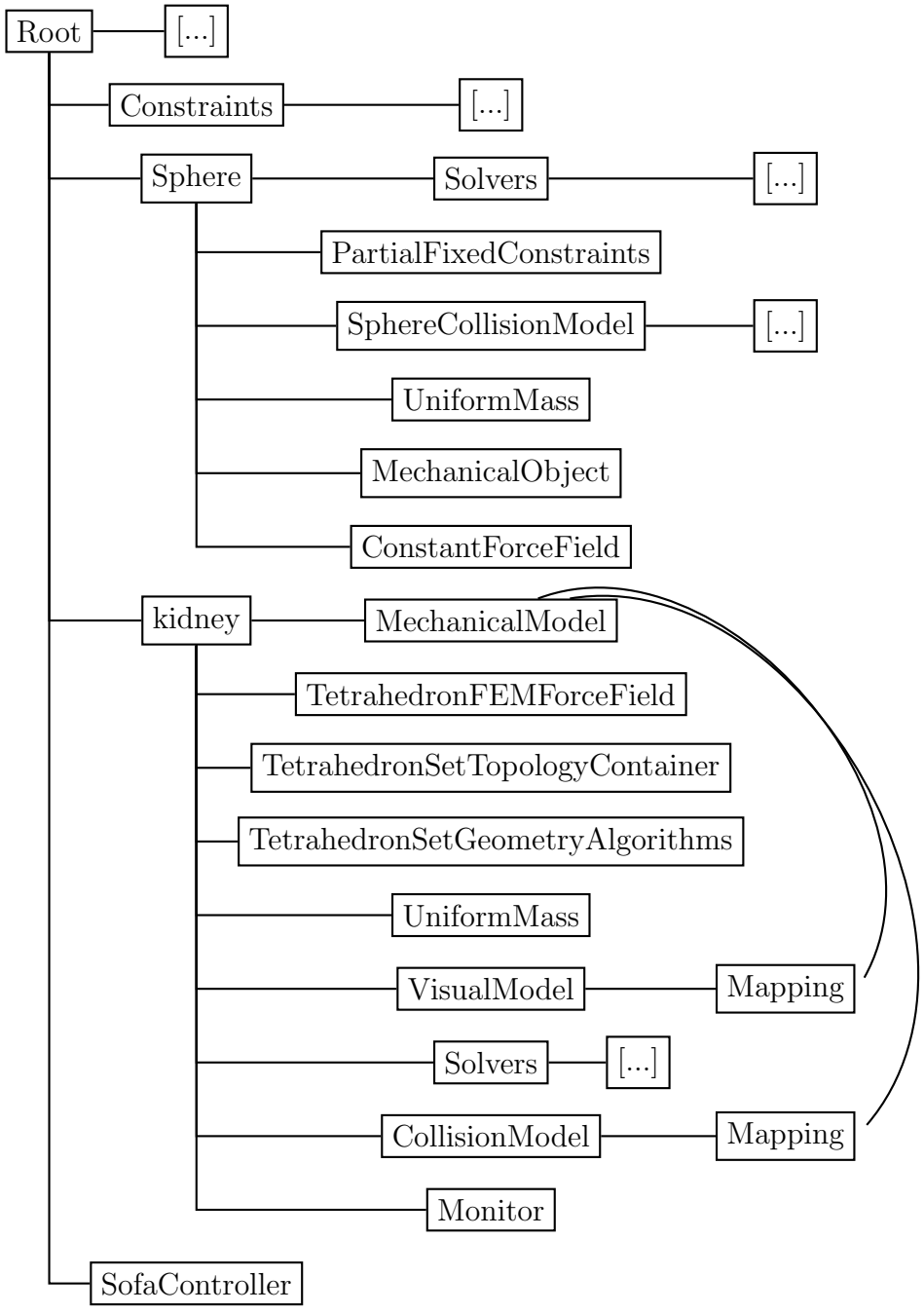


Figure 5.6: Graph of the local deformation simulation

In this simulation two main object are present: *Sphere* and *Kidney*. The main

nodes necessary to the simulation are described in the Section 4.5 (Anatomy of a basic simulation).
For the *Kidney* node: the *MechanicalObject* is a mesh that have been loaded using *MeshLoader*, then we have the three nodes relative to the topology of the mesh used (*TetrahedronFEMForceField*, *TetrahedronSetTopologyContainer*, *TetrahedronSetGeometryAlgorithms*, described in Section 4.5), also necessary to setup the FEM simulation: the Soft tissue material parameters for the kidney can be found in Karimi and Shojaei (2017).
The mechanical, visual and collision model are referred to the same mesh and are connected together using Mappings; the *Monitor* node will keep track of some interesting points of the mesh that have been chosen.
The *Sphere* node contains only the *CollisionModel*: this collision model is, in fact, a sphere of 1 [cm] of radius. It is represented by a rigid particle ad *MechanicalModel*: being a rigid object, i can control its $[x, y, z]$ position in the environment from the *SofaController* and apply forces to it using the *Constant-ForceField* node, that allows to set external forces $[Fx, Fy, Fz]$ to be applied to the particle.
The *PartialFixexConstraint* node allow to block the motion of the particle along one or more of the axes: this allows to control the sphere position perpendicularly to the kidney's surface and avoid sliding.
The result of the simulation will be a deformable body, the kidney, fixed in the environment: is possible to interact with it by controlling a sphere in positon and to apply force with it. By making the sphere colliding with the kidney, the latter will deformate depending on the amount of force applied to it.
In Figure 5.7 is possible to notice the interaction between the sphere and the kidney.

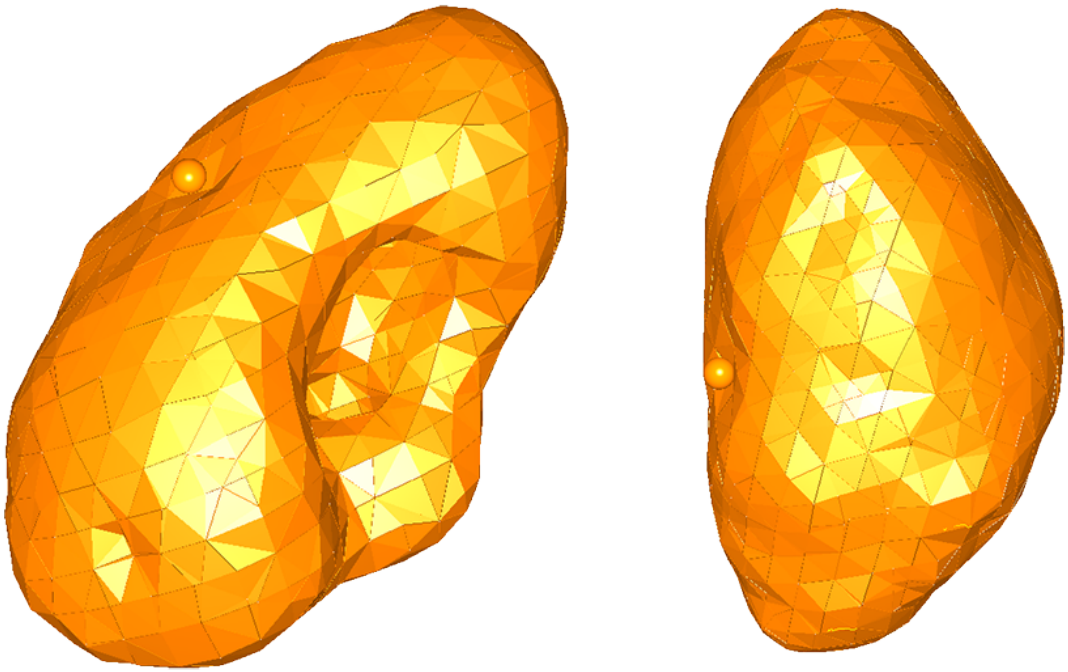


Figure 5.7: Interaction between the kidney and the sphere from two different point of view

Chapter 6

The software development

A big advantages of the SOFA platform are its openness and flexibility due to the possibility to use a Python script to define the simulation and easily interact with it.
During the thesis time, i developed various Python scripts that has allowed me to validate and build various aspects of the simulation and the whole architecture around it (GitHub repository for the code, 2021).

SofaPython SofaPython is a plugin for SOFA: this plugin allows to interact with different aspects of the simulation using a Python script as a controller.
This plugin allows various entry points, each one linked with particular events. At each entry point an associated pre-defined function is defined, is then possible to add some code to it that will be executed when that kind of event happens.
For my needs i used the following entry points:

- *OnLoaded*(): is called when the controller script has been loaded;
- *InitGraph*(): is called when the complete scene has been initialized;
- *onBeginAnimationStep*(): is called at the beginning of each time step;
- *onEndAnimationStep*(): is called when the simulation step ends;

- *onKeyPressed*(): is called when a key has been pressed.

This plugin allows for a direct and handy way to interact with the simulation; i used the first two entry point to initialize the variables and object, in particular the second is very useful to initialize variables after all the object have finished their initialization, being able to record the actual initial values (for instance the position or velocities) before any motion begins.
The third and fourth allow for being able to read or modify the state of the object before or after the simulation step has started or ended; I used those to make an external script interact with the simulation, being able to send commands precisely when the step is about to begin.
The last one allows for a "per-key" input control: is possible to associate at each key of the PC's keyboard or mouse a different function. For instance, a key could increase the position of an object on the x axis of a precise amount, increase the rotation or the force applied or even to reset its initial position (all examples that I implemented in my work).

6.1 The software architecture

In the figure 6.1 is presented an overview of the complete architecture that I have implemented. The *SOFA* component represent the whole simulation program: the *Simulation* sub-node is the actual simulation while *SofaController* is the Python script that i wrote to interact with the simulation and that grants access to the simulation using the SofaPython plugin for SOFA described in the previous section.

In the *onBeginAnimationStep* entry point, called at the beginning of each simulation step, I implemented a bi-directional socket with a Python script external to SOFA. The payload of the socket is a JSON file: this makes the communication easy to implement and very versatile, because the internal format of the JSON file could be modified but the structure of the socket itself won't change;

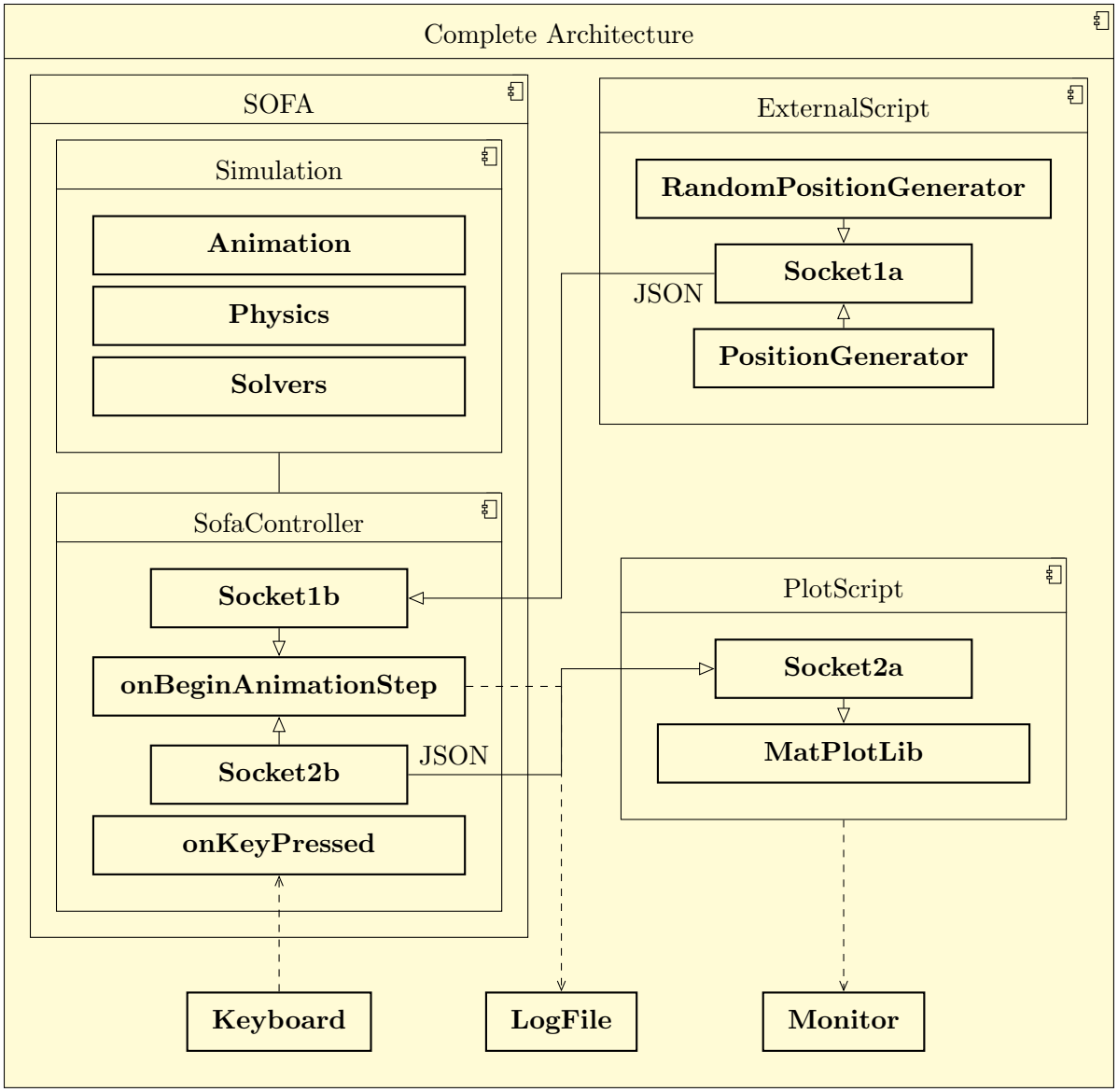


Figure 6.1: Scheme of the complete architecture

additionally, there are libraries integrated in Python to deal with Sockets and JSON files.
The external script gives commands to SofaController: during my work i made a lot of different controls for different aspects, thanks to the versatility of the communication using JSON changing the kind of data that was transferred was quite easy and fast, once the base communication aspect was taken care of.
From the external script would so be possible to control many of the aspects of the simulation and read data from it, making the necessary changes on both the SofaController script and the external script. For instance, is possible to control the position of a point in the simulation, the amount of $ExternalForce$ applied to it, its rotation (for the rigid bodies) and read data such as the position of the point or the position, velocity or force on all the points in a mesh representing an object.
Both the SofaController and ExternalScript programs are able to log the status of some object or other information such as the simulation timestamp onto a text file (not necessarily the same file), again in the JSON format for an easier parsability, readability and re usability of the log at a later time. A new file is being generated each time the simulation or the external script are being launched with a name that includes a timestamp of the launch, to avoid overwriting.
The entry point $onKeyPressed$ is used when the user presses the combination of CTRL and any other key: is possible to define a 'per-key' specific code to run whenever that happens.

6.1.1 Plot the state during the simulation

The *PlotScript* component is again written in Python; it uses a socket in a very similar way to the one described for $ExternalScript$. Here i used the MatPlotLib library to draw graphs: this allow to have a visualization on how the position, velocity or force may change during the simulation time. The graph will output on a window in the PC's monitor and will give information about the state of the objects that are being simulated in simulation time while this is running.

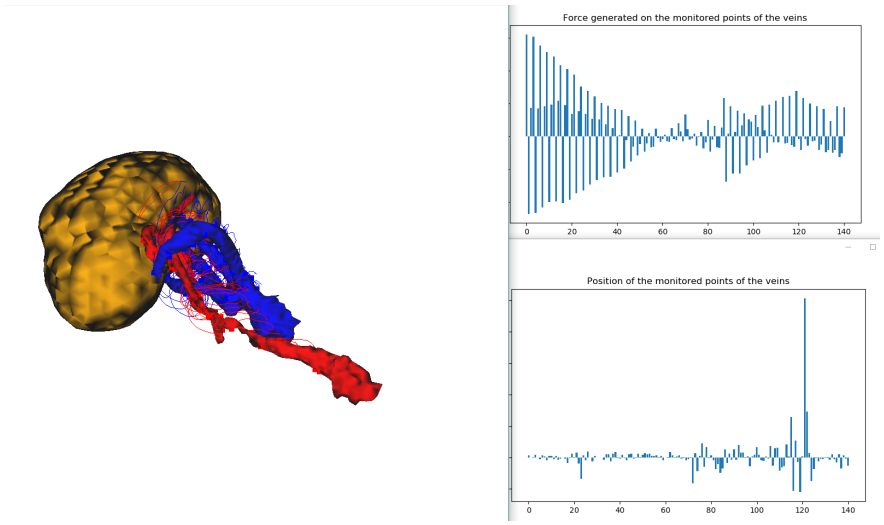


Figure 6.2: Example of graphs plotted during the simulation

6.1.2 Data Parsing and analysis

The data collected from the simulation is logged in text files using a JSON format; using this kind of format will allow the data to be read easily at any time from another script that can parse the data in some new format that is more suitable as input to the Neural Network or in other format standard like CSV.
I built another component to read the recorded data from the log file and than plot it: this allow the user to check if the recorded dataset contains some issues or unexpected values.

6.2 Kidney's blood vessels deformation simulation

In the simulation of the deformation of the blood vessels, the external script is responsible of generating the new position and orientation of the Kindey's me-

chanical model. This will happen by applying a certain ratio-of-change of position and orientation on all three the axes, $[x, y, z]$. The ratio-of-change could be specified manually or generated randomly: in both cases, the ratio will be referred to a time period of 1 [s].
The simulation will communicate the simulation time at each step, together with the current position and orientation of the kidney; the rate-of-change will be applied to those, depending on the chosen time step (in this case is 0.1[s]). The computed pose and rotation will be communicated back to the Sofa Controller script; the socket communication happens in the *onBeginAnimationStep* entry point, so at the beginning of each time step. This means that the change in pose or orientation will be applied at the beginning of each simulation step. In this way, the External script know the amount of time that have passed from a message to another and can compute the correct change to apply at the kidney.
At the end of the animation step, the current state of the kidney, the simulation timestamp and the state of all the monitored points of the Artery and Veins are logged in a text file. Both the socket communication and the file logging happens using JSON formatted file.
If enabled, the randomly generated rate-of-change (and its relative position) will change every 3[s] of simulation time: this allow for a semi-realistic continuous change in pose and orientation over time. For both rotation and translation, upper and lower bounds are specified for each axis: especially in the random situation, this avoid rotation of the kidney that could be dangerous or that cannot be performed during a real surgery.
Additionally, there is also the possibility to control directly the Kidney's pose and orientation using keyboard key: each key can change the amount of rotation's degrees or translation distance separately for each axis.
The *PlotScript* will visualize position, force or velocity (or all three) of the point chosen to be monitored; those will be represented in separate bar graphs, for both the simulated objects.

6.3 Kidney's local deformation simulation

In the simulation of the local deformation, the external script is responsible of control position and orientation of the sphere: the latter will mimic the tip of an instrument of the daVinci robot. The tip will collide with the surface of the kidney and impress a deformation on it.
The simulation will communicate the simulation time at each step: the external script is can be controlled manually, by defining the starting point of the sphere in the direction of the organ's surface.
The computed pose and rotation will be communicated back to the Sofa Controller script; the socket communication happens in the *onBeginAnimationStep* entry point, so at the beginning of each time step. This means that the change in position and force for the sphere will be applied at the beginning of each simulation step. When a new position and force for the sphere have been generated, the organ deformation will reset to the initial state, to not be affected by the previous one.
At the end of the animation step, the initial position of the ball, the current force, the position of the three points in the kidney's mesh closer to the deformation and the simulation timestamp are logged in a text file. Both the socket communication and the file logging happens using JSON formatted file.
The external script can generate random starting point and relative starting force for the sphere: this is done each 4 [s] of simulation time, to give the force generated by the deformation enough time to stabilize. Additionally, there is also the possibility to control directly the position and force applied on the sphere for the sphere using keyboard key: each key can change the amount of force directed or the position separately for each axis .

Chapter 7

Prediction model

7.1 Neural networks

A Neural Network is in general a graph composed of multiple *neurons* connected with each other, generally in connected ordered layers, each one associated with an array of *weights*. The neurons in NN comes conceptually from the human neurons: like the natural ones, they can be connected in groups composing a network.
The neurons will process the input data and return some output: the input data will be weighted by some coefficients (Weights), then the Activation function will generate the output depending on the weighted inputs. The activation function is usually a nonlinear function that will map the sum of the weighted input to a single output.

The learning phase (or training) is a crucial phase because it will define the weights of all the neurons that composes the network, based on the a dataset composed by some input and known output data; each sample will change all the weights of a certain amount. This change can be a really small quantity or be more relevant, depending on the input data itself and the algorithm used. Many techniques can be used, such as the Gradient Descend that uses back-propagation of the error. In some implementation of NNs, the weight can also change while

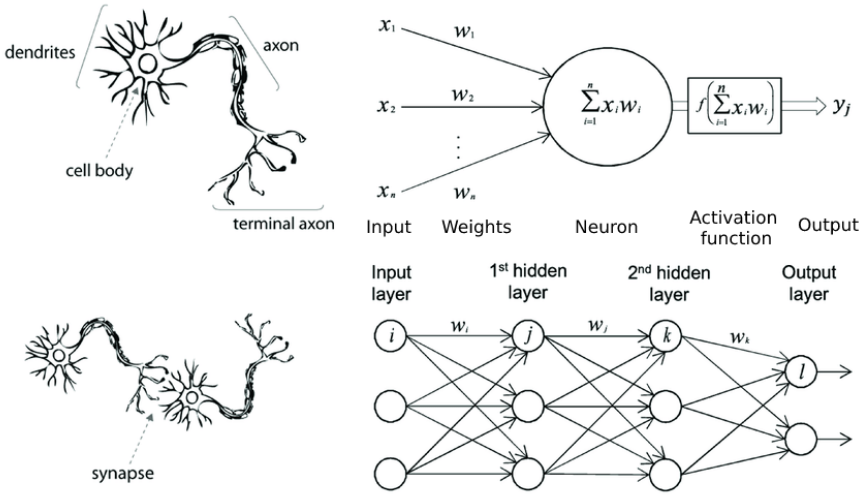


Figure 7.1: An artificial neuron compared with an human one

the network is being used (online learning), when new data, not present in the training dataset, passes through the network.
A network is composed by layers, each one have a certain amount of neurons:

- Input layer: is the first layer, it has as many neurons as the size of the feature input vector (the number of columns of the input data set).
- Hidden layer: those are the layers between the input and the output layer. Is possible to have many of those inside a network but the more of them are present the more complex the training phase will become. Many different kinds of those layers exists.
- Output layer: is the last layer, its output will be the one of the neural network and will represent the final classification, having as many units as the feature output vector.

A trained network can then be used to make predictions when given an input: based on the network, the input data will flow along the layers of neurons, each output of the layers will depend by the input given from the previous layer, to the final output layer when the final result is given.

7.2 Preparation of the data

As i described in Section 6.1 "The software architecture", during the simulation some data relative to some point of particular interest is generated and then saved on a Log file using the JSON format. If we want to use that data to train a model, it has to be parsed in such a way that a Neural Network can take it as an input. A Neural Network needs two main block of information: the input data set, that contains all the data that will be used and the related output data set. The input data set will be used in conjunction with the output data set for training phase. After that, the input data set can be used for validation, comparing the output of the model with the original dataset.
The data set is composed of multiple samples: each sample is identified by a certain number of output features, in this case the position and force of each point of the simulation (belonging to the blood vessels) that is being monitored. The input features associated to each sample are in this case composed by the position and orientation of the kidney that moves (ideally randomly): the position and force of each monitored point of the blood vessels will so be associated to position and orientation of the kidney.
Both output and input feature data set are composed by raw matrices: the first has as many rows as the number of collected samples and as many columns as the number of features; the second has as many rows as the number of collected samples and as many columns as the position and orientation data.
The input data from the log file will contain a lot of different samples and more data than necessary: a script has to remove the unnecessary data, transform the content into a row matrix and split the data and label set into two different files (one for the data and one for the labels).
Both the datasets are then normalized in the range $[0, 1]$ to help the learning phase.

7.3 Neural Networks in Python

To build a neural network is possible to use two very powerful framework, Keras and Tensorflow, which fully supports Python and are very intuitive to use.
They will allow the use of the data sets generated from the simulation, after being prepared, to build a fitting model based on it.

Tensorflow Tensorflow is a library, available for Python and other programming languages, released by Google, that can be used to create deep learning models. It offers a backplane for fast numerical computing, being the kind of computation in machine learning composed of operation on matrices of big size. Tensorflow contains multiple ad-hoc implementation to support different HW capabilities, for instance the use of AVX2, AVX512 or CUDA (when available), to achieve faster computation time, for instance in the learning or inference process. The computation in Tensorflow is described in terms of data flow:

- Node: a node perform a computation, can have zero or multiple input/output;
- Edges: will define the flow of data between different nodes;
- Operation: abstract computation, represented by a directed graph;
- Data: represented in term of multi-dimensional arrays of real values.

Tensorflow is so a very powerful mathematical library that can be used as a backbone for a machine learning model implementation.

Keras Keras is a library for Python that can run on top on Tensorflow (in fact, in the last versions of Tensorflow's Python implementation Keras was integrated into Tensorflow itself).
The creator of Keras had those principles in mind:

- Modularity: allow for the model to be composed as a sequence of graphs or just a single one, use deep learning models as discrete components that can be combined in multiple ways;
- Minimalism: the library provides just the necessary tools to achieve an outcome;
- Extensibility: new components can be added and created easily to add new features;
- Python: this library was created with Python in mind and for this reason is completely built using this language.

Keras was built to make the creation and development of model that uses deep learning faster and easier, without having to use Tensorflow directly. It provides API for the mathematical operations that are needed for deep learning, providing interfaces called *backends*.
The main model type in Keras is called *Sequential*, a linear stack of multiple layers, from the input to the output ones, with one or more levels in the middle. To build and use a model in Keras is necessary to perform those steps:

1. Define the model: create a Sequential model and add the necessary layers;
2. Compile the model: specify loss function and apply various optimizers;
3. Fit the model: train the model on a dataset;
4. Make prediction and evaluate the model: generate some prediction and validate them by comparing the expected value with the generated one.

7.4 Generating the data

As described in Section 6.1 (The software architecture), the data generated from the simulation, regarding the position and forces of the points of the 3D models that are being tracked, in addition to other information, is being written onto a

log file while the simulation is running with a sampling rate of 10 [Hz].
Recording the simulation running for approximately 400 [s] of simulation time, using the method described in Section 5.1 to generate random linear and angular velocity for the kidney, will generate a time sequence of position and forces of the points and position and rotation of the kidney. From each of those sequence I will build a dataset to then be used to train or validate the model. The measurement unit for the linear distances will be the *centimeter* while the one for the rotation will be *arcdegree*.
Each sample of the datasets has this shape:

$$[x, y, z, R_x, R_y, R_z, x_1, y_1, z_1, \ ..., x_n, y_n, z_n]$$

7.4.1 Training sets

I generated a total of 17 dataset that will be normalized and then used to train the neural network.

	Dataset name	Number of samples
1	0920114247	4642
2	0920121748	4464
3	0920125053	4272
4	0927225115	4870
5	0927232519	4101
6	0927235241	4136
7	0928113454	4374
8	0928120640	4065
9	0930134844	4625
10	0930142045	4101
11	0930144850	4103
12	0930151638	4030
13	0930174546	4783
14	0930184441	4212
15	0930195935	4168
16	0930203905	4589
17	0930212107	4761

Table 7.1: Training datasets

7.4.2 Test sets

I generated a total of three test set to test the behavior of the network, using the same metodology used for generating the training sets.

	Dataset name	Number of samples
1	0927221920	4392
2	0929110030	4382
3	1013115727	4156

Table 7.2: Test datasets

7.5 Training and inference pipeline

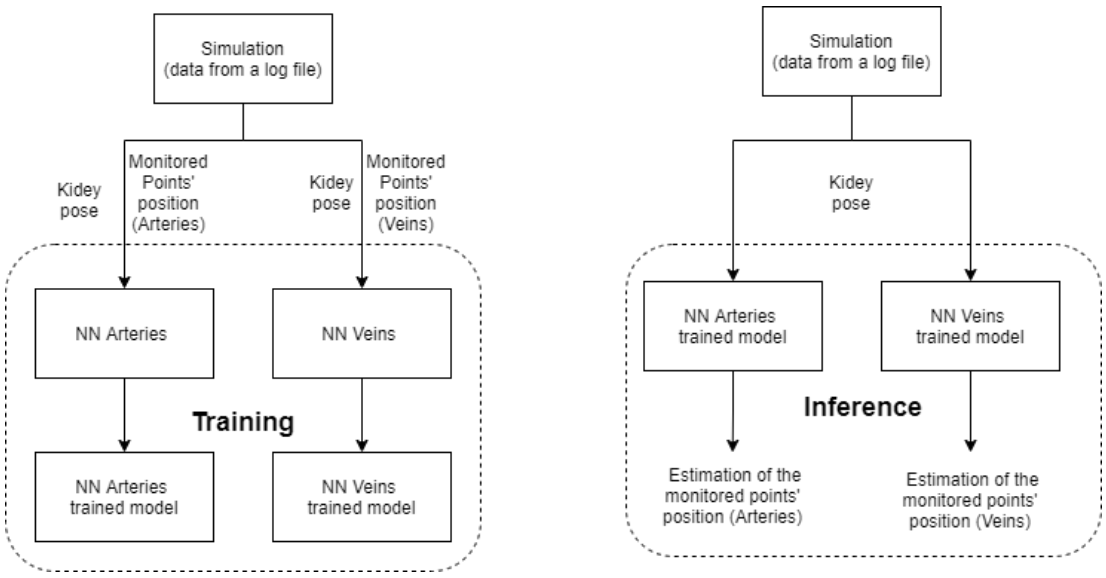


Figure 7.2: Training and inference pipeline of the Neural Networks

In the Figure 7.2 I present the pipelines for the training and inference phases. In the training phase, i give as input the position and rotation of the kidney and as reference output the position of the monitored points; I will build two neural networks, one for the monitored points of the veins and one for the arteries separately.

In the inference phase, as input the position and rotation of the kidney is given and the networks will return the estimation of the position of the monitored points, for both arteries and veins. The quality of the estimation can than be compared with the actual position of the point of the blood vessel to understand how good the prediction is.

7.6 Build the models

7.6.1 Regression Neural Network

The first NN that I tried for my purposes is a Multi-layer Perceptron for Multi-Output regression: the Perceptron is a single neuron model, precursor of the modern deeper and larger networks that are composed of multiple and more complex Perceptron-like layers. Each input is directly connected with the first level (the input layer) and then a dense hidden layer: dense means fully connected, all the neurons from the input layer are connected to each one of the neurons in the hidden layer and the same can be said for the neurons in the hidden layer in relation with the ones in the output layer.

```
# function to generate the model
def get_model(n_inputs, n_outputs):
  model = Sequential()
  # hidden layer
  model.add(Dense(50, input_dim=n_inputs,
      kernel_initializer='he_uniform', activation='relu'))
  # output layer
  model.add(Dense(n_outputs))
  model.compile(loss='mae', optimizer='adam')
  return model
```

The hidden layer in this case is composed by 50 neurons. The size of the output layer will depend by the amount of features that are present in the data sets: in my experiments I used different size of the output vector, having used the $[x, y, z]$

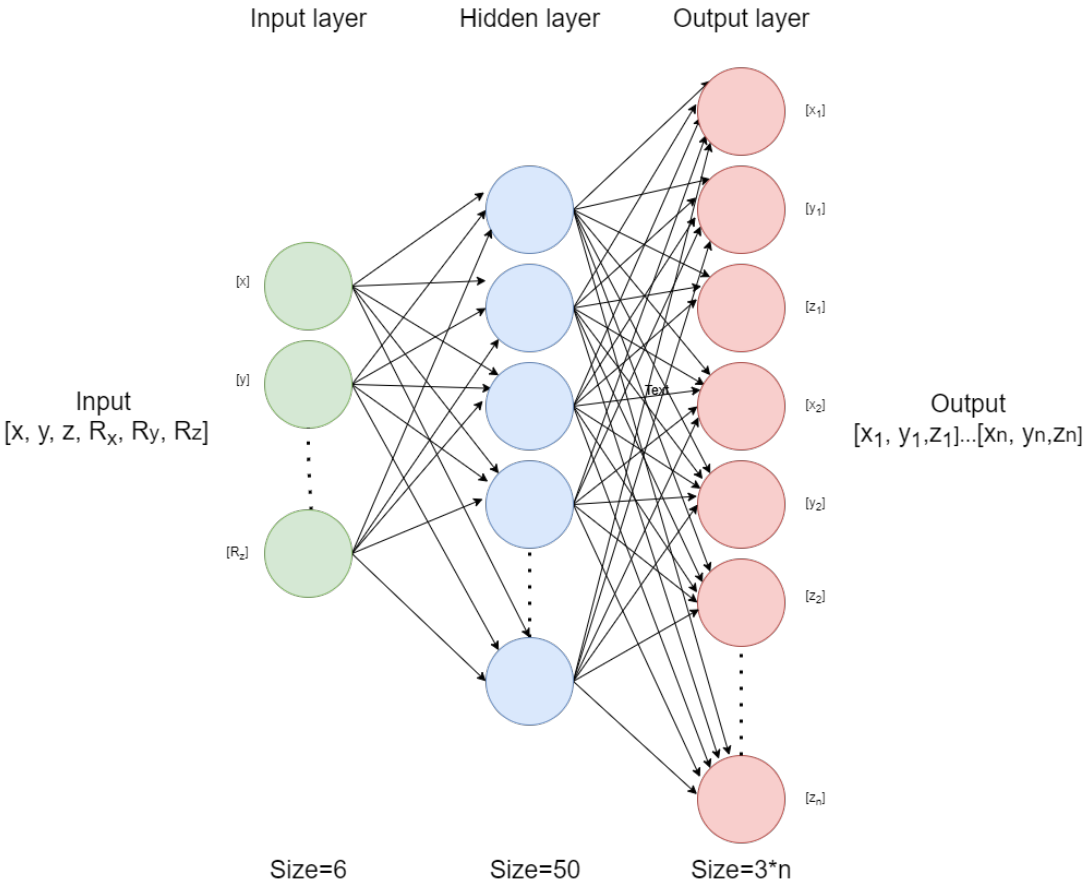


Figure 7.3: Scheme of the implemented regression neural network

positions of the various monitored points (described in Section 5.1, "Simulating the blood vessels' deformation") of the Vein or Artery simulated objects (that vary in number of points), or the position in addition to the force directed in the three axes (in addition to the three position the dataset will contain also the three forces, for each of the monitored points). The size of the output will so be $[3 * n]$ if we're using the position of the monitored points or $[6 * n]$, if we're using both position and forces; n in this case is the number of points that we are tracking from the simulation model. In the case of the veins, we will have 54 points, in the case of the arteries 47. The size of the input one will always remain constant and of size [6], three features for the $[x, y, z]$ position and three for the rotation

along the axes, $[R_x, R_y, R_x]$.
To build the model i used the *ReLu* (Rectified Linear Unit) as activation function for the neurons in the hidden layer: this is the most popular activation function, Glorot et al. (2011) discovered that it allows for a better and faster training of the network with respect to other activation functions, such as the logistic sigmoid or the hyperbolic tangent.
The loss function used for this model is the *mean squared error*, which is the one that is generally used for regression problems.
As optimizer, *adam* was chosen: the name comes from adaptive moment estimation; Adam is a replacement optimization algorithm for stochastic gradient descent for training neural networks which has good performance and is considered the best overall choice by Ruder (2017).

7.6.2 LSTM Neural Network

The previous neural network, while simple, is very powerful; still, there is an information that that kind of network cannot exploit: time. Being in my case the datasets composed of time sequences of poses of a kidney, I can use a kind of neural network that is able to use the temporal information and use not just the current input sample but also the ones before to infer the current output.
The LSTM (or Long Short-Term memory) is a Recurrent Neural Network (RNN) architecture: recurrent in this case means that, differently from the standard neural networks, this kind of NN is able to process entire sequences of data, having feedback connection between the cells that bring the information regarding previous sequences.
I used a multiple-to-one approach using a moving window (Figure 7.4): 30 subsequent samples of a sequence (corresponding to 3 [s] of simulation time, being the sampling time 0.1 [s]) are used as input to train the network and, again, 30 subsequent samples to infer the actual value.

The input for this neural network is not just a matrix composed by multiple samples, each one with columns for the features: it becomes a 3D tensor, each input sample is composed by a matrix containing 30 samples of the kidney's pose:

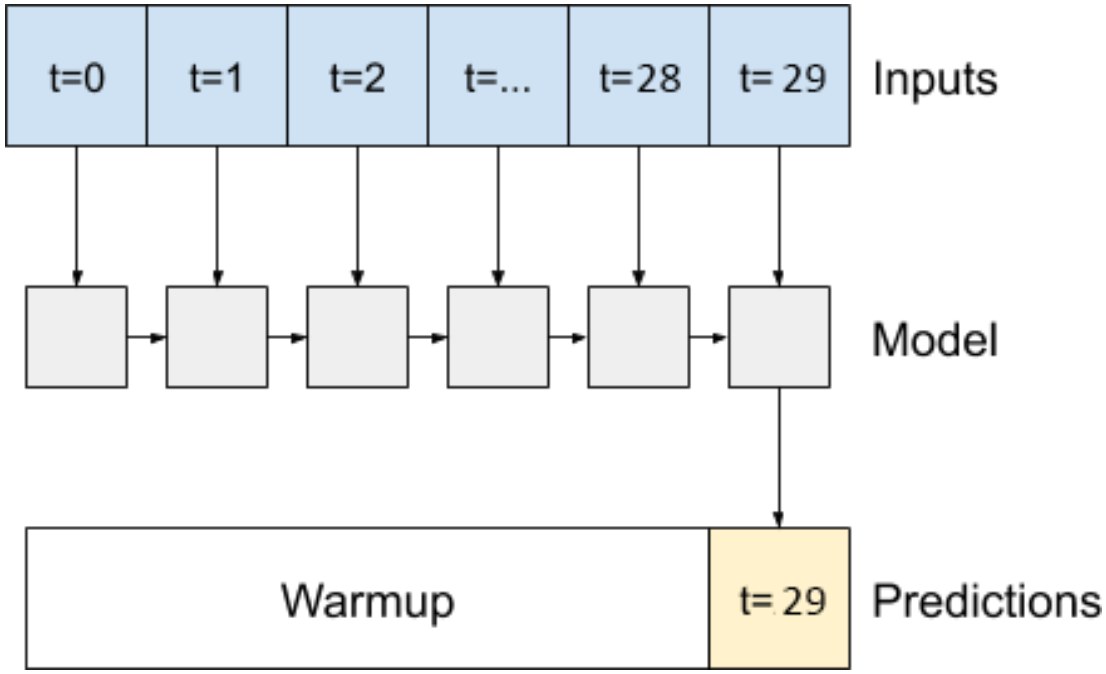


Figure 7.4: Moving window as input of the LSTM network

the current pose of the kidney and the 29 poses of the previous samples. The output of the network will be the current estimated position of the points in the veins/arteries, similarly to the previously explained network.

An LSTM unit is composed by a cell and three gates: input, output and forget gate. The cell is the one that remembers the values and the three gates regulates the flow of information that comes through the cell.

The information arrives to the forget gate: this gate decides if the current value has to be discarded or kept.

Then the value arrives to the input gate after being subjected to a sigmoid activation function: this gate will decide how important the information will be: the cell will then record the value, it will be much or less relevant, depending on how important the input gate has reported it to be.

The output gate then reports what the next state will be, it contains information from the previous states and can also be used for predictions. At each state, the previous hidden state and the current input are concatenated and from them the

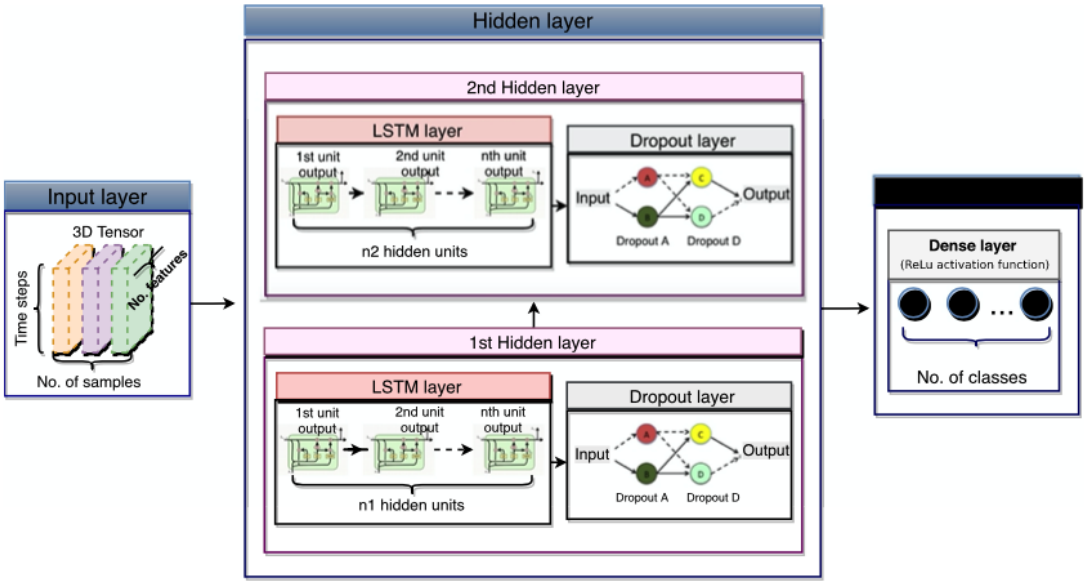


Figure 7.5: Scheme of the implemented LSTM network

output is generated.

The structure of the implemented network is the following:

```
def get_model(n_inputs, n_outputs):
  model = tf.keras.Sequential()
  # First LSTM layer
  model.add(layers.LSTM(128, input_shape=(None, n_inputs),
    return_sequences=True))
  model.add(layers.Dropout(0.2))
  # Second LSTM layer
  model.add(layers.LSTM(128, return_sequences=False))
  # Dense hidden layer
  model.add(Dense(64, activation='relu'))
  model.add(layers.Dropout(0.2))
  # Output layer
  model.add(Dense(n_outputs, activation='relu'))
  model.compile(loss="mean_absolute_error",
    optimizer='adam')
  return model
```

Similarly to the regression neural network, I used the same optimizer and loss function.
I used an LSTM level which contains 128 units; the level will take as input 30 sequences; it will return the same sequences to the second LSTM layer (with the same number of units) after being subjected to a Dropout layer. The second layer will not pass the sequences but a single vector as output. The data passes then to a Dense layer of size 64 (with ReLu as activation function), then a Dropout and the final Dense layer that will constitute the output layer.

Chapter 8

Analysis of the results

To validate the results, I trained the networks by using the training dataset listed in Section 7.4.1 and then used the dataset for test listed in the section 7.4.2.
I built a script to infer a whole test dataset and, at the end, plot graphs that compares the evolution of the position for each point in $[x, y, z]$ between the simulation results and the output of the model.
The regression model resulted in a decent accuracy in the predictions but the quality of the estimation was not the same for all the point: some points are correctly tracked but other points are not. The model built using LSTM layers have a slightly better accuracy overall but, especially for some points that where troublesome for the first network, the error is lower and the predicted evolution of the position for most of the points is much similar to the one that comes from the simulation.
I then built a script that allows to visually compare the result of the model and the data that comes from the simulation, having them side by side. They will follow the same instants of time and same position and orientation of the kidney as input for both the simulation and the model.

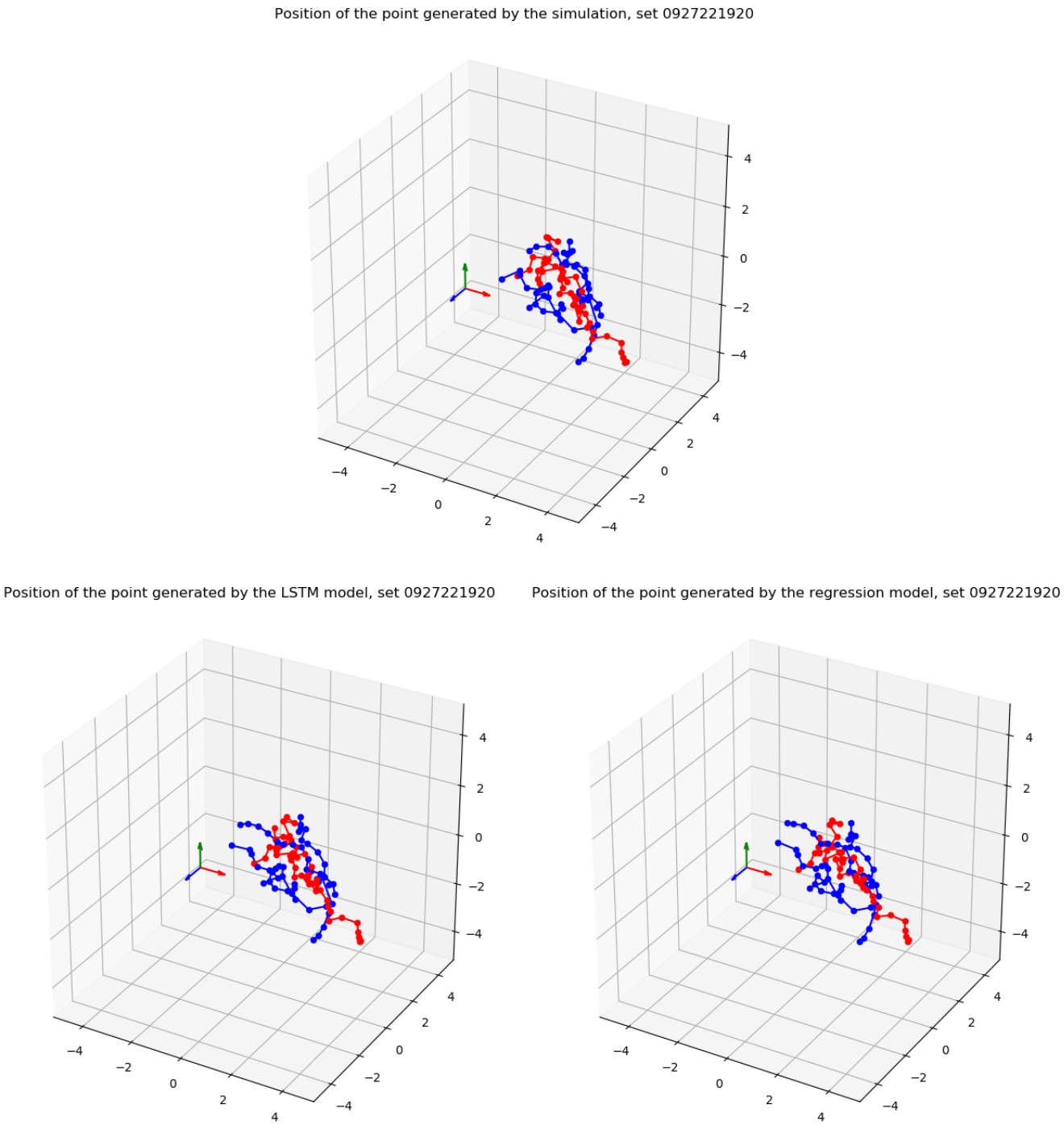


Figure 8.1: Comparing the prediction of the models with the simulation output

8.1 The monitored points

From the results that is possible to observe from the visual output of the scripts (the graph of the positions), both models are able to give a reasonable output with respect to the results of the simulation. As a reminder, the features that I keep in account are the $[x, y, z]$ position of some handpicked points of the blood vessels. While the simulation runs, the position of those points is logged and, when the model is used, those are the features that are going to be predicted. In the Figure 8.2 is possible to see the points that are being monitored for the *veins*. Looking at the results of the *vein*'s neural network (but simular conclusion could

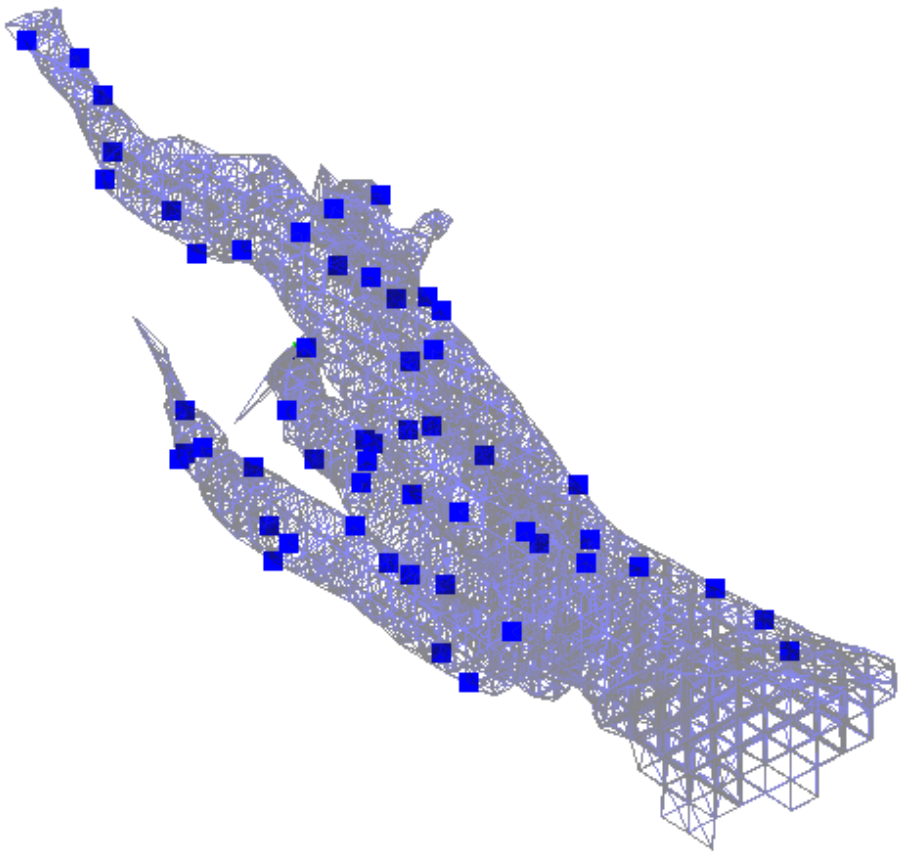


Figure 8.2: Monitored points for the *veins*

be drown for the *arteries* one), is possible to observe that some points are being tracked in a worse way than others: this behavior is usually consistent among the datasets and the models. The points that are connected to the kidney or in its neighborhoods are predicted very accurately, similar thing happens for the point near the Vena Cava (the opposite side). This is probably due to the fact that the points in between oscillate a lot while moving and the model is trying to compute a mean value between the oscillations.

In the Figure 8.3 is possible to see a classification that I made from the results

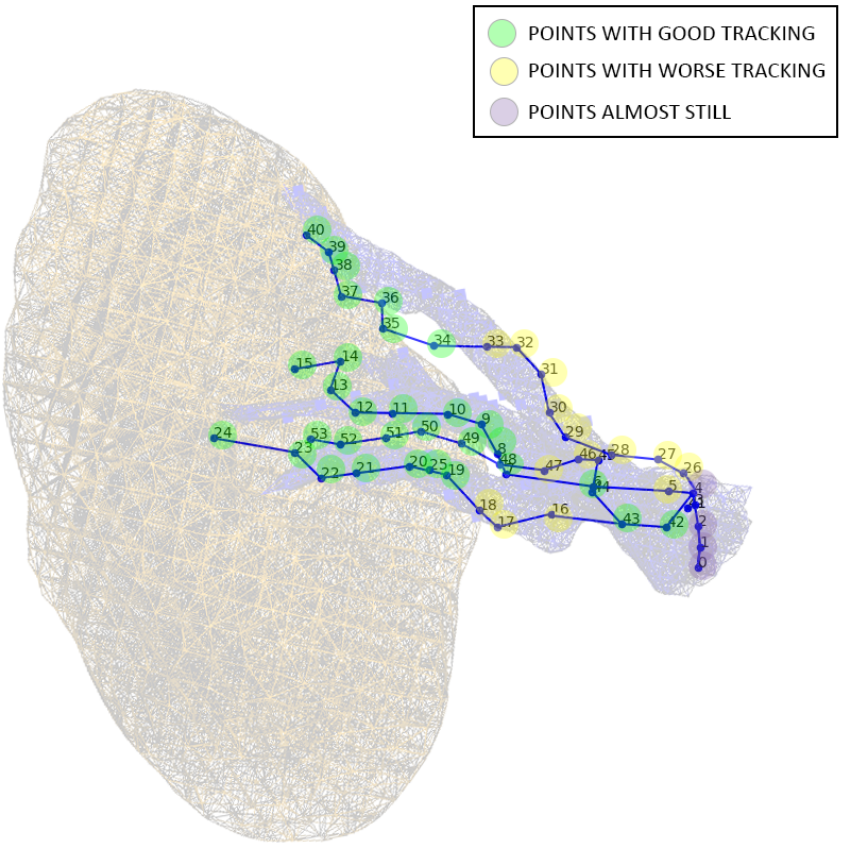


Figure 8.3: Classification of the monitored points for the *veins*

in the inference of the test datasets: the point in green are the ones that can be

considered as *good* tracked, while the one in yellow are the ones that i considered *worse* tracked. In addition, there are some points that i considered *still*, they're very close to the fixed constraint (on the Vena Cava) and for that reason they do not move much, not just when inferencing from the model but also in the simulation.

In general, among the datasets, is possible to observe that some lower or higher peaks are not correctly predicted, especially in the points that i considered *worse*. While for the *good* points the difference in tracking between the regression model and the LSTM network is quite similar (they both behave well), the same cannot be said for the *worse* points: the regression model is not very effective in predicting their positions while the LSTM works quite better.

I will report here some of the points of the "0927221920" training dataset (for the veins' points) comparing the results of the LSTM and the regression model.

For the point #11 is possible to observe that the while the regression model is able to follow generally the trajectory, it fails to reach some of the peaks that are instead reached (up to a certain amount) by the LSTM model.

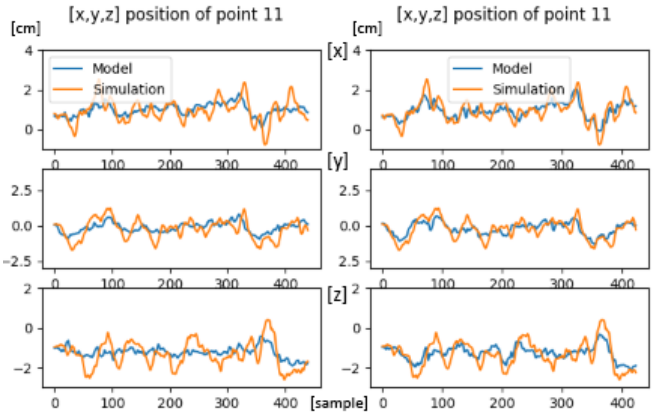


Figure 8.4: Prediction of Point #11 using the Regression model (left) and the LSTM (right)

For the point #12, is possible to make similar observations to the ones for the point #11: the trajectory is generally tracked well by the LSTM, peaks included. The point #31 is one of the worse tracked point of the dataset, and this is very

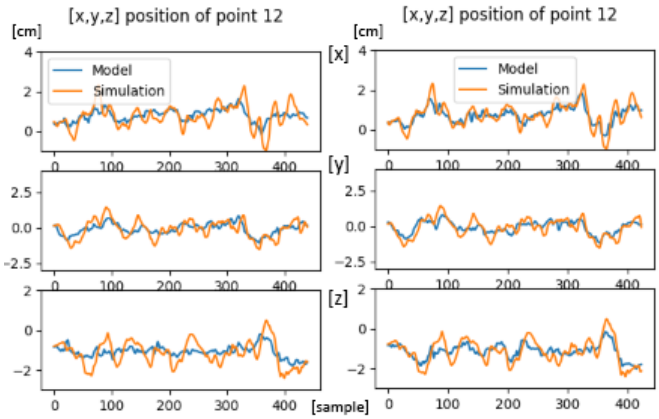


Figure 8.5: Prediction of Point #12 using the Regression model (left) and the LSTM (right)

evident if we look the result of the regression model. The LSTM works better, especially on the y and z axes.

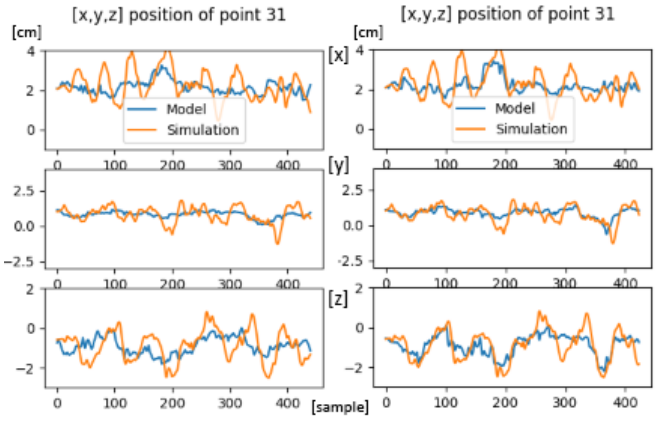


Figure 8.6: Prediction of Point #31 using the Regression model (left) and the LSTM (right)

Again, for the point #32 we can draw similar conclusions: the LSTM works better, especially in the peaks.

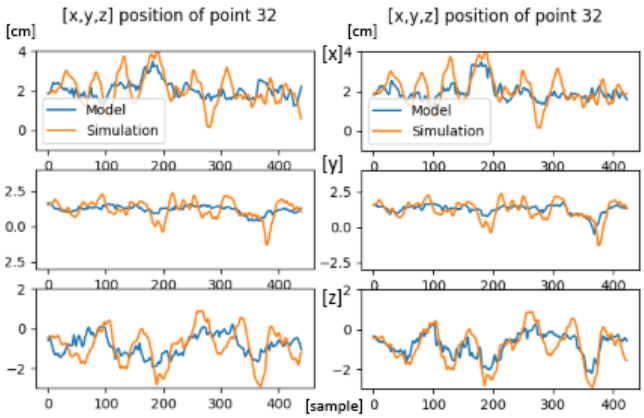


Figure 8.7: Prediction of Point #32 using the Regression model (left) and the LSTM (right)

The point #38 is one of the best tracked points: the result from the regression model is quite good but the one that comes from the LSTM is even better.

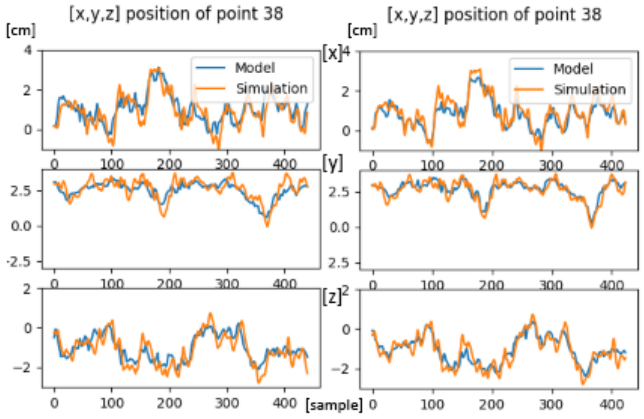


Figure 8.8: Prediction of Point #38 using the Regression model (left) and the LSTM (right)

Similarly to the point #38, the point #50 has a nice tracking in the LSTM over all the sequence.

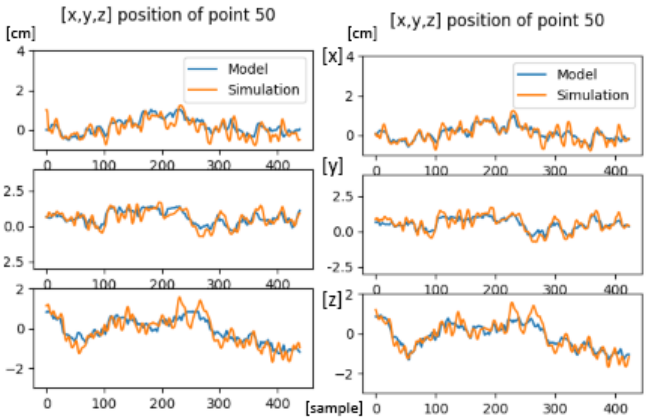


Figure 8.9: Prediction of Point #50 using the Regression model (left) and the LSTM (right)

8.2 Comparing the models using metrics

To compare the results between the two models and also between the models and the original sequence i used three metrics: the mean Gaussian distance, the Pearson Correlation Coefficient and the Dynamic Time Warping (DTW).
For the Gaussian distance, I computed the mean distance between each point of the original sequence and the predicted one axis by axis, then the final mean Gaussian distance between the points, expressed in centimeters.
The Pearson correlation index is a measure of linear association between two sets of data; in particular, is the ratio between the covariance of two variables and the product of their standard deviations. The result of the computation will return a value $P \in [-1, 1]$: 1 means that the sequences are positive correlated, -1 means that the sequences are negative correlated, while 0 means that there is no correlation at all. For my purposes, I would like a value that is as close as possible to 1. To compute this index, I have reshaped the time sequences of the

dataset (two dimensional, being time sample of the position of multiple points) into a 1D array, then I used the *scipy* implementation (Scipy is a Python Open Source library offering mathematical instruments and algoritm).
The DTW is a method to measure the similarity between multiple time series: it is invariant to time warping and this makes it able to find correlation that wouldn't be evident when using the mean Gaussian distance. To compute it I used the *fastDTW* library for Python, using a window of 30 samples and *euclidean* as the method to measure the distance.
I will compute those metrics for each of the test dataset between the original sequence and the inferred one, then the result from the different models will be compared.

8.2.1 Analysis on the test set '0927221920', veins

To give a better understanding of the effectiveness of the models and the difference between them, just for this dataset, I reported in Table 8.1 containing the mean distance between the output of the simulation and the prediction of the models, among the three axes for each single point. The distance is expressed in centimeter: is possible to appreciate the fact that the LSTM has an error that is marginally better than the regression model. The average value for the axes are: $[0.295, 0.3245, 0.319]$ for the Regression model and $[0.244, 0.27, 0.257]$ for the LSTM.
The maximum error can be found for the points 31-35 for both models: those points moves quite a lot during the simulation so we can expect an higher error.
We can then consider the Gaussian distance in average of all points: for the Regression model, it averages to 0.626 while for the LSTM is 0.514 (the lower, the better).
The Pearson correlation index between the simulation results and the Regression model is 0.955, comparing it with the LSTM the index becomes 0.969 (the higher, the better).
For the DTW, the distance between Regression model and simulation is 22437.56, the distance between simulation and LSTM is 18430.49 (the lower, the better).

	Regression Model			LSTM				Regression Model			LSTM		
#	x	y	z	x	y	z	#	x	y	z	x	y	z
0	0.050	0.073	0.052	0.047	0.066	0.045	27	0.293	0.172	0.291	0.250	0.149	0.219
1	0.072	0.078	0.063	0.062	0.070	0.056	28	0.317	0.231	0.364	0.278	0.199	0.276
2	0.110	0.087	0.083	0.096	0.078	0.073	29	0.363	0.295	0.449	0.325	0.255	0.348
3	0.159	0.101	0.113	0.133	0.088	0.100	30	0.466	0.347	0.518	0.409	0.293	0.381
4	0.192	0.107	0.147	0.159	0.095	0.121	31	0.550	0.395	0.614	0.472	0.336	0.415
5	0.193	0.133	0.143	0.157	0.111	0.118	32	0.553	0.455	0.660	0.446	0.371	0.437
6	0.196	0.234	0.176	0.160	0.189	0.135	33	0.550	0.478	0.641	0.446	0.411	0.443
7	0.255	0.388	0.298	0.211	0.318	0.220	34	0.596	0.536	0.605	0.481	0.460	0.450
8	0.333	0.420	0.375	0.274	0.356	0.287	35	0.572	0.514	0.550	0.469	0.412	0.394
9	0.369	0.487	0.453	0.299	0.407	0.346	36	0.534	0.557	0.447	0.421	0.431	0.346
10	0.365	0.536	0.480	0.295	0.433	0.362	37	0.494	0.481	0.423	0.380	0.343	0.309
11	0.361	0.444	0.466	0.287	0.357	0.366	38	0.393	0.404	0.348	0.299	0.270	0.265
12	0.342	0.365	0.417	0.264	0.317	0.317	39	0.312	0.309	0.276	0.258	0.212	0.236
13	0.283	0.339	0.347	0.233	0.300	0.244	40	0.259	0.235	0.223	0.293	0.204	0.248
14	0.327	0.354	0.349	0.295	0.309	0.264	41	0.197	0.142	0.170	0.166	0.119	0.126
15	0.265	0.211	0.186	0.204	0.225	0.179	42	0.223	0.231	0.173	0.183	0.193	0.131
16	0.171	0.259	0.228	0.153	0.214	0.170	43	0.209	0.253	0.200	0.177	0.208	0.155
17	0.232	0.368	0.291	0.212	0.289	0.238	44	0.226	0.244	0.219	0.182	0.218	0.185
18	0.250	0.411	0.294	0.222	0.325	0.242	45	0.262	0.280	0.308	0.208	0.248	0.257
19	0.246	0.433	0.324	0.215	0.371	0.283	46	0.278	0.336	0.321	0.216	0.264	0.278
20	0.266	0.456	0.337	0.229	0.413	0.307	47	0.274	0.355	0.290	0.211	0.278	0.259
21	0.295	0.469	0.360	0.240	0.397	0.345	48	0.269	0.299	0.276	0.200	0.250	0.247
22	0.281	0.490	0.364	0.230	0.426	0.368	49	0.255	0.269	0.285	0.206	0.238	0.236
23	0.247	0.496	0.344	0.218	0.436	0.335	50	0.243	0.280	0.297	0.181	0.239	0.238
24	0.281	0.385	0.335	0.253	0.328	0.310	51	0.231	0.277	0.270	0.169	0.223	0.241
25	0.252	0.444	0.328	0.220	0.389	0.294	52	0.216	0.237	0.250	0.153	0.185	0.255
26	0.247	0.134	0.217	0.205	0.121	0.167	53	0.179	0.201	0.226	0.150	0.185	0.244

Table 8.1: Comparing the distance from the points in the simulation to the predicted ones between the Regression model and the LSTM

8.2.2 Summary table for the veins datasets

		Regression	LSTM
927221920	Mean dist. [x, y, x]	[0.295, 0.3245, 0.319]	[0.244, 0.270, 0.257]
	Mean Gaus. Dist.	0.626	0.514
	Pearson index	0.955	0.969
	DTW distance	22437.56	18430.49
929110030	Mean dist. [x, y, x]	[0.319, 0.364 , 0.425]	[0.236, 0.294, 0.312]
	Mean Gaus. Dist.	0.736	0.564
	Pearson index	0.930	0.961
	DTW distance	26407.92	19954.57
1013115727	Mean dist. [x, y, x]	[0.326, 0.386, 0.397]	[0.335, 0.396, 0.418]
	Mean Gaus. Dist.	0.735	0.765
	Pearson index	0.923	0.920
	DTW distance	25196.87	25483.59

Table 8.2: Metrics of the vein's datasets

8.2.3 Summary table for the arteries datasets

		Regression	LSTM
927221920	Mean dist. [x, y, x]	[0.493, 0.482, 0.419]	[0.441, 0.453, 0.403]
	Mean Gaus. Dist.	0.927	0.861
	Pearson index	0.959	0.964
	DTW distance	30595.99	27530,58
929110030	Mean dist. [x, y, x]	[0.462, 0.557, 0.646]	[0.423, 0.533, 0.585]
	Mean Gaus. Dist.	1.119	1.029
	Pearson index	0.935	0.944
	DTW distance	36546.24	33132,93
1013115727	Mean dist. [x, y, x]	[0.549, 0.612, 0.702]	[0.599, 0.607 , 0.751]
	Mean Gaus. Dist.	1.242	1.296
	Pearson index	0.924	0.916
	DTW distance	38211.77	38463.96

Table 8.3: Metrics of the arteries' datasets

8.3 Concluding remarks on the models

The metrics that I proposed confirm what is possible to see visually from the graph representing the evolution of the position of the points: the LSTM model usually outperforms the Regression model. This is by far more evident while analyzing the result for the *veins*: in this case, the Regression model is definitely less effective and the LSTM is able to perform much better in all the proposed test dataset.

Is interesting to note that the LSTM has usually better performance than the Regression model, except for the '1013115727' set: in this case the performance will regress on most of the proposed metrics, for both the *veins* and *arteries* datasets.

After the conclusions, it can be found an appendix chapter (Additional graphs) containing the graph for the position for all the points of some datasets so it would be possible to compare all of them point by point.

I consider the results good and very promising for future works: the models have to generate an output composed by 162 or 141 (veins or arteries datasets, $[x_1, y_1, z_1, \ ..., x_n, y_n, z_n,]$ position of the points) from an input of just 6 features ($[x, y, z, R_x, R_y, R_z]$ of the kidney) and such is a quite complex task.

The neural networks that I implemented are not too complex from the structure standpoint, I think that the result could possibly improve if a more complex Neural Network is used.

Additional, those networks are very sensitive to the amount of training data they have at disposal: by doubling the training set (from 8 to 17) I observed that the results have improved by a small but significant margin, that could be observed just by looking at the graph of the evolution of the points.

Chapter 9

Conclusion

This study has been an initial analysis that would bring to the final objective of building an augmented surgery application. This is one of the most recent and advanced field of studies in Robotic surgery: the ability to track the deformation on human organs during surgery is a very complex task, due to the constraints in space and the necessity of being as minimally obtrusive as possible (ideally not obtrusive at all).
We have found the estimation of the deformation of the blood vessel attached to a kidney a very interesting problem to analyze, having the possibility to measure the pose of the kidney. Being able to estimate the position of the blood vessel, it would be then possible to give the surgeon information about their deformation, if they are in a dangerous position or even to show on the visor offered by a robot such as the daVinci a 3D model of them super-imposed on the camera stream.
To be able to draw some conclusion has been necessary to build from scratch a simulation, the architecture around it and then some models to compare.
The simulation has been implemented using SOFA and can be personalized from the actual patient: the data used to build the 3D models used in the simulation comes from the CT scan of the patient and, after some processing, can be imported in the simulation. I would then use the simulation as a source for data that i will the use to build a model.

In Figure 9.1 is possible to appreciate the SW architecture and the data flow that I have thought out to be able to build the dataset; those will be then used to train and validate the models. The *ControlScript* connects bidirectionally with the simulation (using the SofaPython plugin): the script will generate random linear and angular velocities that will change the pose of the kidney in the actual simulation. The kidney will so move and because of that the attached blood vessels will deform. The data has then to be logged in a text file; the content of the text file will be used to generate a dataset (for both *veins* and *arteries*). Once a certain amount of dataset have been collected, is then possible to build a model: the datasets will serve now as training data and, when the training process ends, is possible to save the obtained Neural Network to reuse it later.

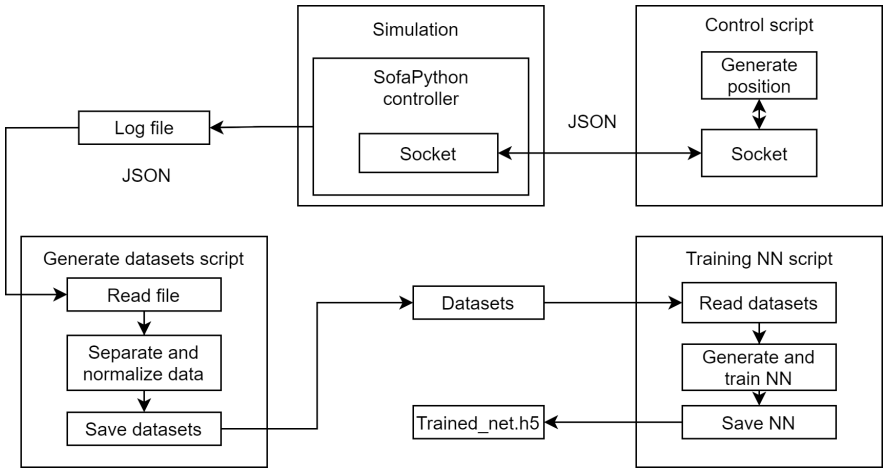


Figure 9.1: Sw architecture to train the model

In Figure 9.2, instead, is shown the SW architecture and the data flow implemented to inference from the model while the simulation is running: the first part is the same as the training phase, the external control script will give linear and angular velocity commands to the simulation to control the pose of the kidney. In this case, the data won't be just saved but it will be also sent to another script. The latter will receive the current state of the simulation (the pose of the kidney and the current position of the monitored points, for both veins and

arteries) and, after having loaded the Neural Network previously trained, it will be able to give the pose of the kidney as input to the network and so predict the position of the chosen points.

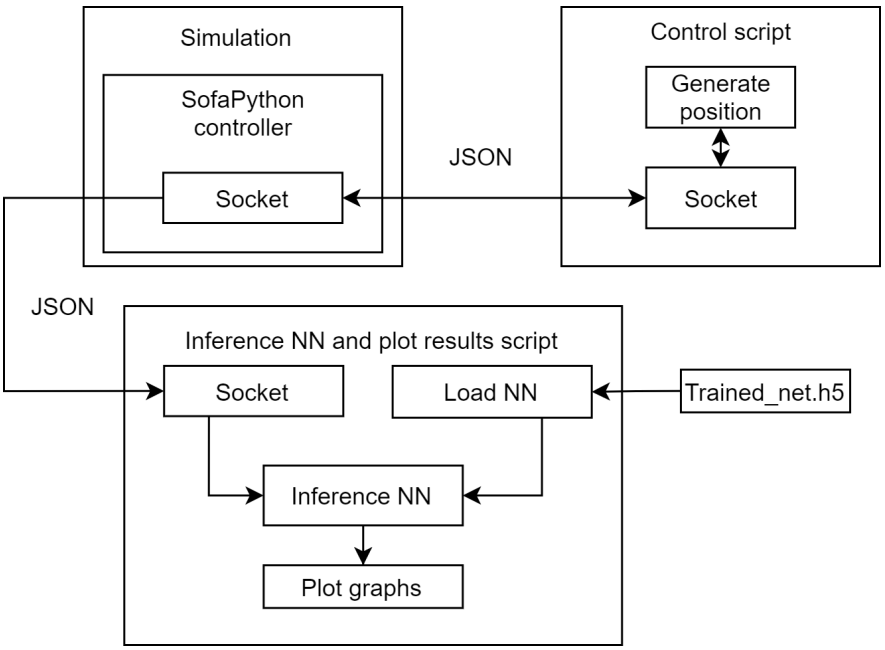


Figure 9.2: Sw architecture to inference from the model

The script has an additional feature: is able to plot in real time the current position of the monitored points side by side with the current position predicted by the model. As can be observed from Figure 9.3, it is so possible to visually compare the simulation, the results of the simulation and the results from the trained model. I added in both the plots an additional element, the indicator of the frame of the kidney (located in its 3D model's barycenter): this will evolve in position and rotation in the same way that the kidney in the simulation does and can be an useful visual feedback representing how the organ is moving.

In principle, it could now be possible to discard the simulation in the inference phase and attach the architecture to a flow of data that comes from an actual organ, once a reliable source for the data will be validated.

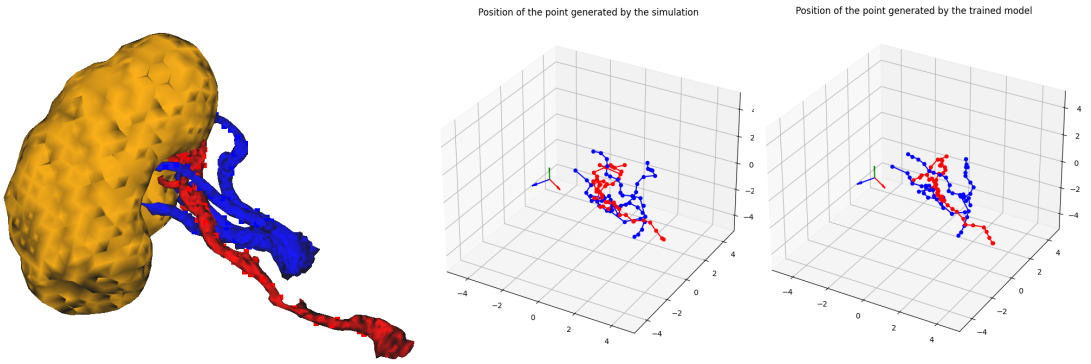


Figure 9.3: Predicting in real time the shape of the blood vessels

The 3D model that I used in the simulation was unfortunately not specific for the use in the simulation but was just for visual support and aid to the surgeon. This means that the model had some geometrical issues that would make the simulation not actually "true to life". The solution to this problem would be to use a model built specifically to be used in the simulation.

To build the augmented surgery application with holograms superimposed on the real organs, is also necessary a way to rebuild the entire organ 3D model from the subset of points available: i picked around 150 points while the 3D model has vertices in order of the thousands. The output of the neural network (when all points are connected) has a shape that is very similar to the 3D original model and for this reason build back the original shape should not be too hard, using interpolation techniques from the available points.
If the goal of the augmented surgery application wouldn't be the graphical representation but, instead, giving contextual information to the surgeon, it would be possible to generate a model with information derived from the previous one to alert him when a dangerous situation could be encountered.

The approach I used was inspired by some of the works that can be found in the literature but studying the deformation of the blood vessels when moving the

kidney and then try to build a model from predicting the deformation is, to my knowledge at least, unique. The obtained results with the complete architecture I designed are promising, there is still a lot that can be made to make it better and usable in the practice.

To conclude, I hope that my contribution will be useful for future works: the augmented surgery can be extremely helpful, especially during complex surgical operations. The possibility of simulating a surgery (or by using a model to avoid the heavy computation necessary for the simulation) can be useful also during the training of future surgeons.

Appendix A

Additional graphs

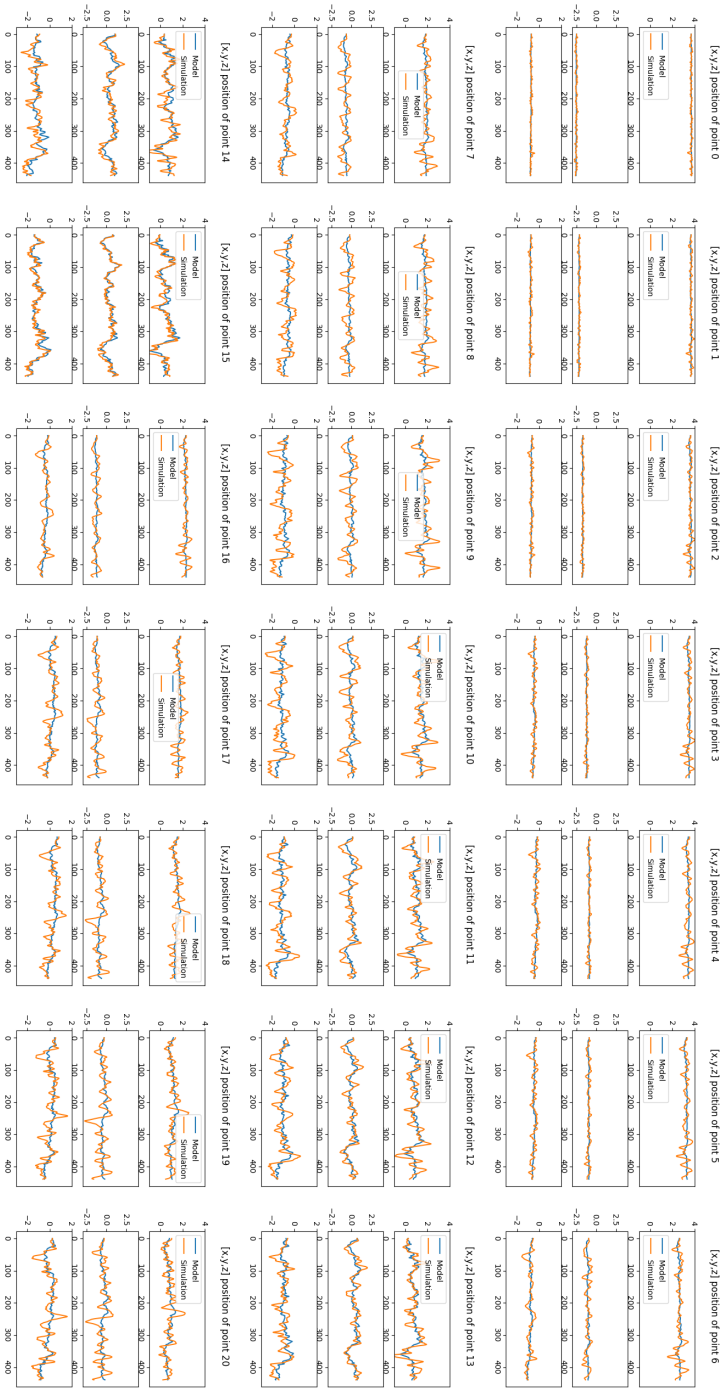


Figure A.1: Output of the Regression model compared with the original sequence (Dataset '027221920', veins)

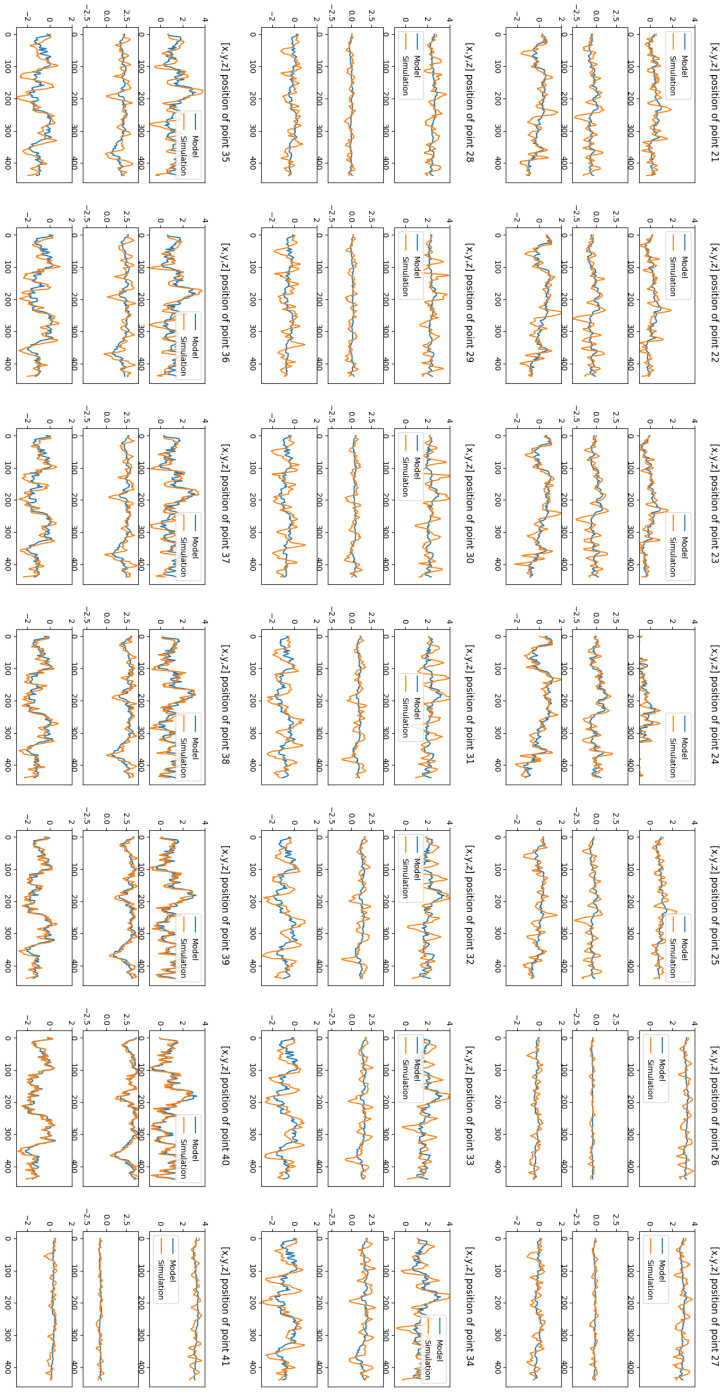


Figure A.2: Output of the Regression model compared with the original sequence (Dataset '027221920', veins)

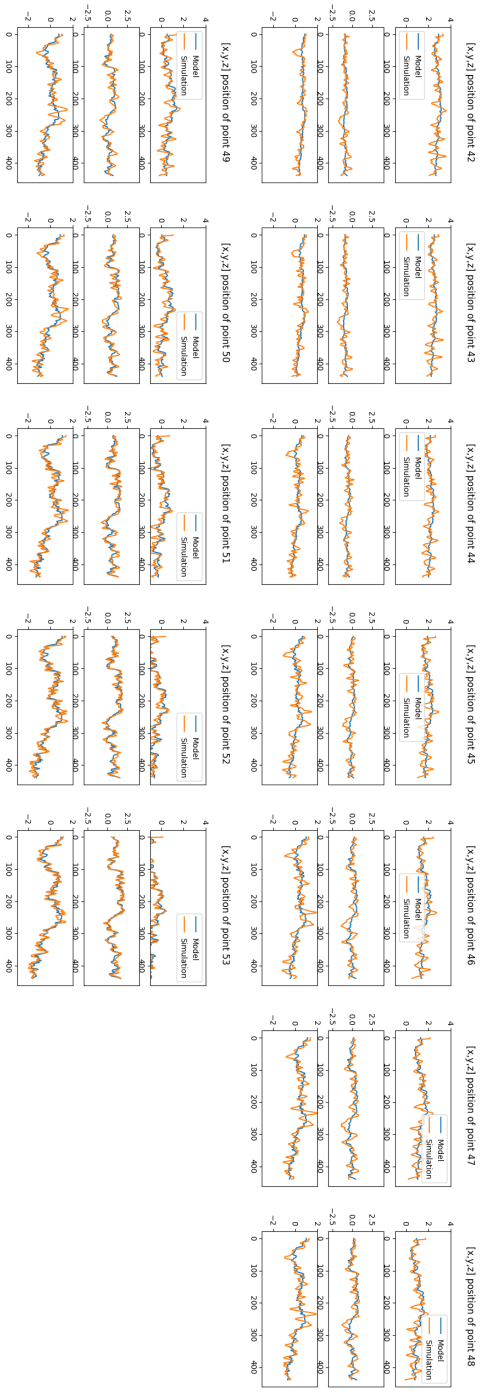


Figure A.3: Output of the Regression model compared with the original sequence (Dataset '027221920', veins)

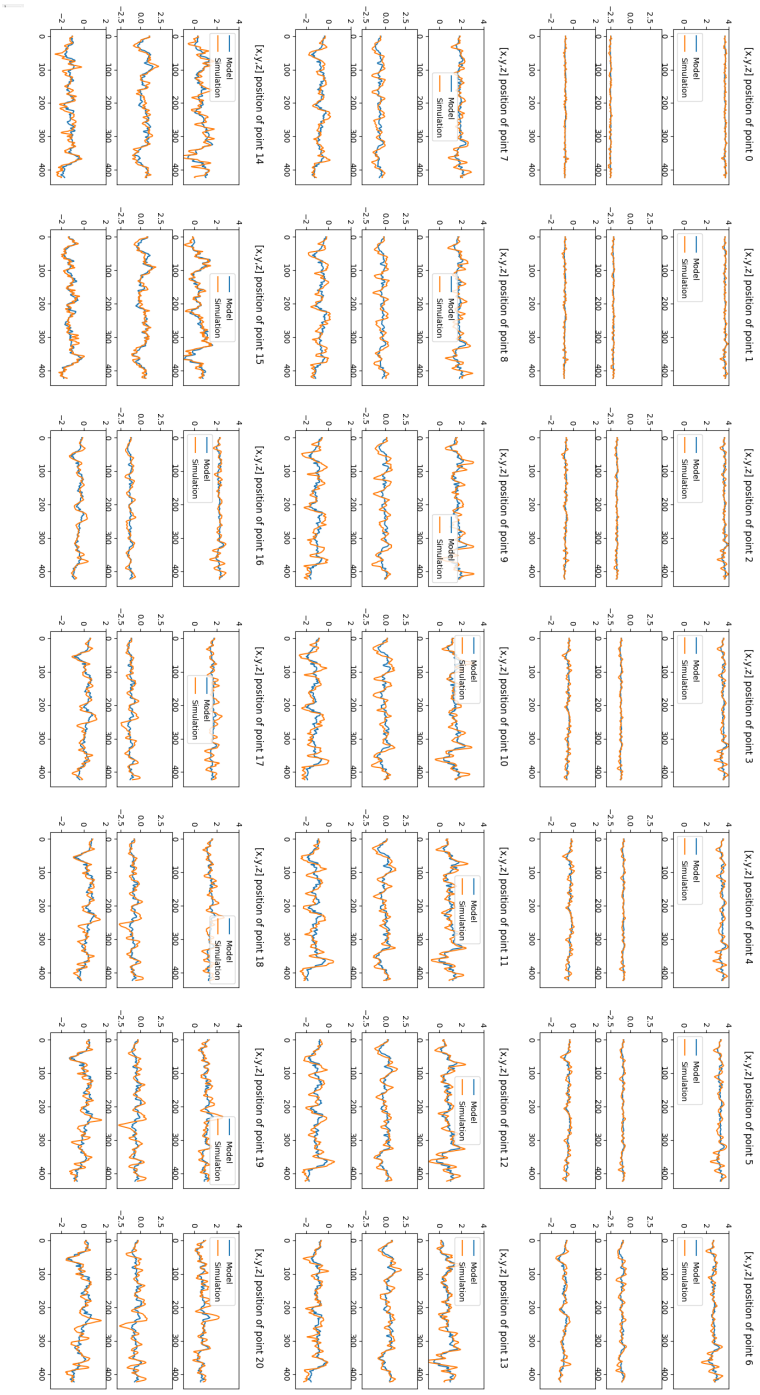


Figure A.4: Output of the LSTM model compared with the original sequence (Dataset '027221920', veins)

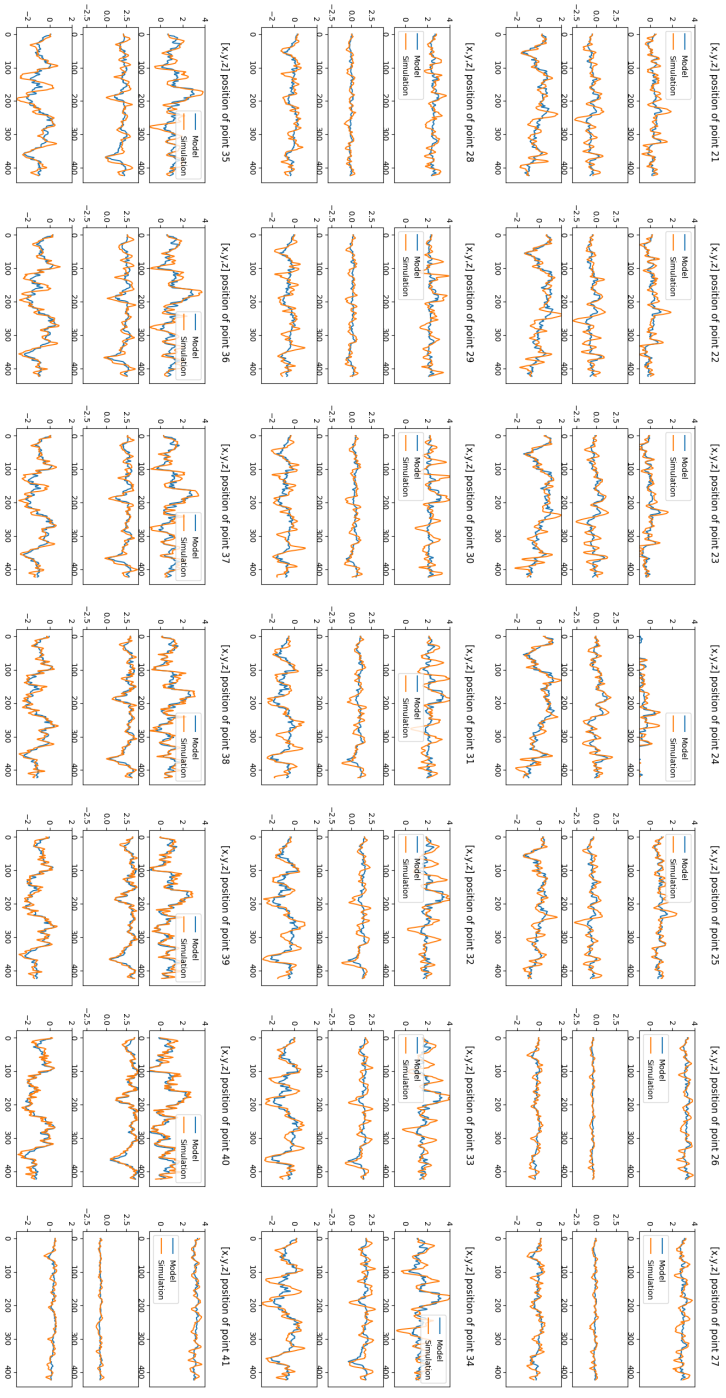


Figure A.5: Output of the LSTM model compared with the original sequence (Dataset '027221920', veins)

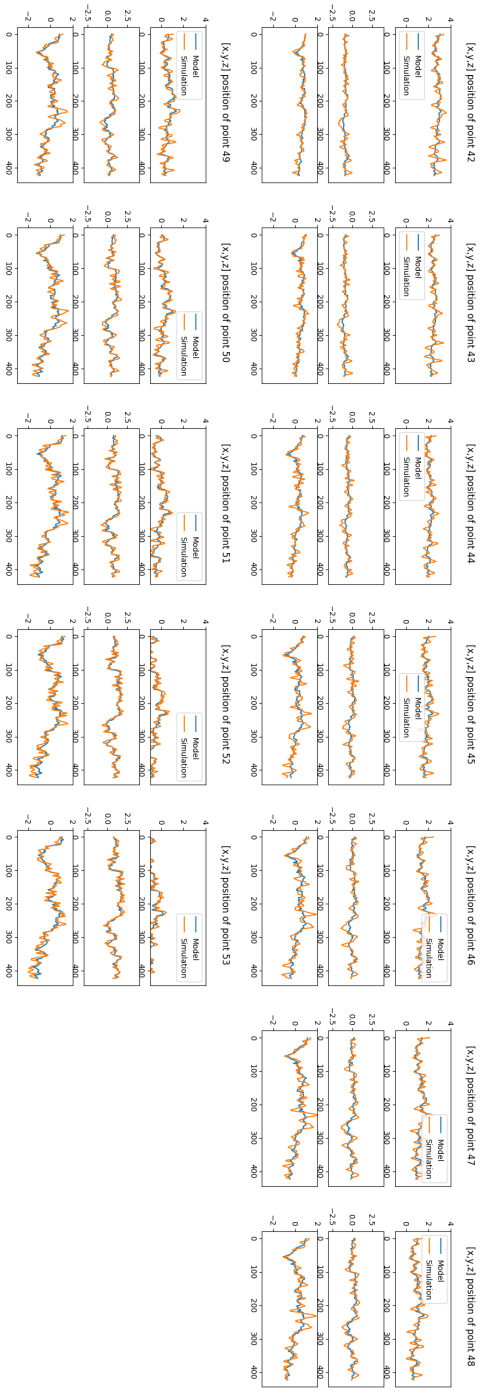


Figure A.6: Output of the LSTM model compared with the original sequence (Dataset '027221920', veins)

References

Abbou, C.-C., Hoznek, A., Salomon, L., Olsson, L. E., Lobontiu, A., Saint, F., Cicco, A., Antiphon, P., and Chopin, D. (2017). Laparoscopic radical prostatectomy with a remote controlled robot. *Journal of Urology*, 197(2S). 11

Ackerman, M. J. (1999). The visible human project. *Academic Medicine*, 74(6):667–70. 29

Allard, J., Cotin, S., Faure, F., Bensoussan, P. ., Poyer, F., Duriez, C., Delingette, H., and Grisoni, L. (2007). *SOFA-an open source framework for medical simulation*, volume 125 of *Studies in Health Technology and Informatics*. PubMed. Cited By :151. 33

Andrew, A. M. (2000). LEVEL SET METHODS AND FAST MARCHING METHODS: EVOLVING INTERFACES IN COMPUTATIONAL GEOMETRY, FLUID MECHANICS, COMPUTER VISION, AND MATERIALS SCIENCE, ISBN (paperback) 0-521-64557-3, (hardback) 0-521-64204-3 (pbk, £18.95). *Robotica*, 18(1):89–92. 29

Ashrafian, H., Clancy, O., Grover, V., and Darzi, A. (2017). The evolution of robotic surgery: surgical and anaesthetic aspects. *British Journal of Anaesthesia*, 119:i72–i84. 4, 6, 7, 15

Bahar, L., Sharon, Y., and Nisky, I. (2020). Surgeon-centered analysis of robot-assisted needle driving under different force feedback conditions. *Frontiers in Neurorobotics*, 13. 15

Bolenz, C., Gupta, A., Hotze, T., Ho, R., Cadeddu, J. A., Roehrborn, C. G., and Lotan, Y. (2010). Cost comparison of robotic, laparoscopic, and open radical prostatectomy for prostate cancer. *European Urology*, 57(3):453–458. 14

Bonet, J. and Wood, R. D. (2008). *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press. 21

Brebbia, C. A., Telles, J. C. F., and Wrobel, L. C. (1984). *Boundary Element Techniques*. Springer Berlin Heidelberg. 23

Bro-Nielsen, M. (1998). Finite element modeling in surgery simulation. *Proceedings of the IEEE*, 86(3):490–503. 25

Camarillo, D. B., Krummel, T. M., and Salisbury, J. (2004). Robotic technology in surgery: Past, present, and future. *The American Journal of Surgery*, 188(4):2–15. 6, 8, 10

Casale, P., Lughezzani, G., Buffi, N., Larcher, A., Porter, J., and Mottrie, A. (2019). Evolution of robot-assisted partial nephrectomy: Techniques and outcomes from the transatlantic robotic nephron-sparing surgery study group. *European Urology*, 76(2):222–227. 12, 13

Chacko, B. and Sawant, H. (2011). Virtual surgery on geometric model of real human organ data. *SasTechJournal*. 29

Chanthasopeephan, T., Desai, J. P., and Lau, A. C. W. (2004). Deformation resistance in soft tissue cutting: a parametric study. In *12th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2004. HAPTICS '04. Proceedings.*, pages 323–330. 23

Chanthasopeephan, T., Desai, J. P., and Lau, A. C. W. (2007). Modeling soft-tissue deformation prior to cutting for surgical simulation: Finite element analysis and study of cutting parameters. *IEEE Transactions on Biomedical Engineering*, 54(3):349–359. 23

Clements, L. W., Dumpuri, P., Chapman, W. C., Dawant, B. M., Galloway, R. L., and Miga, M. I. (2011). Organ surface deformation measurement and analysis in open hepatic surgery: Method and preliminary results from 12 clinical cases. *IEEE Transactions on Biomedical Engineering*, 58(8):2280–2289. 29

Comas, O., Taylor, Z. A., Allard, J., Ourselin, S., Cotin, S., and Passenger, J. (2008). Efficient nonlinear FEM for soft tissue modelling and its GPU implementation within the open source framework SOFA. In *Biomedical Simulation*, pages 28–39. Springer Berlin Heidelberg. 34

Cook, R. D., Malkus, D. S., Plesha, M. E., and Witt, R. J. (2007). *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons, Inc., Hoboken, NJ, USA. 23

Cotin, S., Delingette, H., and Ayache, N. (1996). Real time volumetric deformable models for surgery simulation. In *Lecture Notes in Computer Science*, pages 535–540. Springer Berlin Heidelberg. 21

Cotin, S., Delingette, H., and Ayache, N. (1999). Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):62–73. 25, 26, 29

Crouch, J. R., Pizer, S. M., Chaney, E. L., Hu, Y., Mageras, G. S., and Zaider, M. (2007). Automated finite-element analysis for deformable registration of prostate images. *IEEE Transactions on Medical Imaging*, 26(10):1379–1390. 29

Dawant, B. M., Pan, S., and Li, R. (2001). Robust segmentation of medical images using geometric deformable models and a dynamic speed function. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2001*, pages 1040–1047. Springer Berlin Heidelberg. 29

Deanna, G., Lee, W., Andrew, L., Hawkeye, K., Alicia, C., Thomas, G., Bryan, C., Blake, H., and S., L. T. (2014). Raven surgical robot training in preparation

for da vinci. *Studies in Health Technology and Informatics*, 196(Medicine Meets Virtual Reality 21):135–141. 9

Delingette, H. (1999). General object reconstruction based on simplex meshes. *International Journal of Computer Vision*, 32(2):111–146. 34

DiMaio, S. and Salcudean, S. (2002). Needle insertion modelling and simulation. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*. IEEE. 28

DiMaio, S. and Salcudean, S. (2005). Needle steering and motion planning in soft tissues. *IEEE Transactions on Biomedical Engineering*, 52(6):965–974. 25

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., and Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118. 17

Famaey, N. and Sloten, J. V. (2008). Soft tissue modelling for applications in virtual surgery and surgical robotics. *Computer Methods in Biomechanics and Biomedical Engineering*, 11(4):351–366. 20

Faure, F., Duriez, C., Delingette, H., Allard, J., Gilles, B., Marchesseau, S., Talbot, H., Courtecuisse, H., Bousquet, G., Peterlik, I., and Cotin, S. (2012). *SOFA: A Multi-Model Framework for Interactive Physical Simulation*, volume 11 of *Studies in Mechanobiology, Tissue Engineering and Biomaterials*. Springer. Cited By :138. 34

Favorito, L. A. (2018). Partial nephrectomy: three dimensional (3d) models from preoperative computed tomography is the future to identify the exact location of the tumor. *International braz j urol*, 44(5):857–858. 29

Fung, Y. (1967). Elasticity of soft tissues in simple elongation. *American Journal of Physiology-Legacy Content*, 213(6):1532–1544. 21

Fung, Y.-C. (1993). *Biomechanics*. Springer New York. 20, 21, 28

Galletly, N., McGinty, J., Dunsby, C., Teixeira, F., Requejo-Isidro, J., Munro, I., Elson, D., Neil, M., Chu, A., French, P., and Stamp, G. (2008). Fluorescence lifetime imaging distinguishes basal cell carcinoma from surrounding uninvolved skin. *British Journal of Dermatology*, 159(1):152–161. 16

Gasser, T. C., Ogden, R. W., and Holzapfel, G. A. (2005). Hyperelastic modelling of arterial layers with distributed collagen fibre orientations. *Journal of The Royal Society Interface*, 3(6):15–35. 21

Gelder, A. V. (1998). Approximate simulation of elastic membranes by triangulated spring meshes. *Journal of Graphics Tools*, 3(2):21–41. 27

Gettman, M. T., Blute, M. L., Chow, G. K., Neururer, R., Bartsch, G., and Peschel, R. (2004). Robotic-assisted laparoscopic partial nephrectomy: Technique and initial clinical experience with da vinci robotic system. *Urology*, 64(5):914–918. 12

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA. PMLR. 71

Greengard, L. and Rokhlin, V. (1997). A new version of the fast multipole method for the laplace equation in three dimensions. *Acta Numerica*, 6:229–269. 22

Hammond, F. L., Howe, R. D., and Wood, R. J. (2013). Dexterous high-precision robotic wrist for micromanipulation. In *2013 16th International Conference on Advanced Robotics (ICAR)*. IEEE. 17

Hekman, M. C., Rijpkema, M., Langenhuijsen, J. F., Boerman, O. C., Oosterwijk, E., and Mulders, P. F. (2018). Intraoperative imaging techniques to support complete tumor resection in partial nephrectomy. *European Urology Focus*, 4(6):960–968. 16

Hibbeler, R. C. (2005). *Structural Analysis*. Prentice Hall, 6 edition. 21

Hollenstein, M., Jabareen, M., Breitenstein, S., Riener, M.-O., Clavien, P.-A., Bajka, M., and Mazza, E. (2009). Intraoperative mechanical characterization of human liver. *PAMM*, 9(1):83–86. 28

Holzapfel, G. A. and Weizsäcker, H. W. (1998). Biomechanical behavior of the arterial wall and its numerical characterization. *Computers in Biology and Medicine*, 28(4):377–392. 22

Horgan, C. O. and Saccomandi, G. (2003). Finite thermoelasticity with limiting chain extensibility. *Journal of the Mechanics and Physics of Solids*, 51(6):1127–1146. 21

Horn, M., Reh, B., Wenz, F., Stallkamp, J., and Mombaur, K. (2016). Patient specific corotated FEM simulation and gelatin phantom for prostate brachytherapy. In *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*. IEEE. 38

Hrennikoff, A. (1941). Solution of problems of elasticity by the framework method. *Journal of Applied Mechanics*, 8(4):A169–A175. 26

Hu, T. and Desai, J. P. (2004). Characterization of soft-tissue material properties: Large deformation analysis. In *Medical Simulation*, pages 28–37. Springer Berlin Heidelberg. 27

Hughes, D., Camp, C., O'Hara, J., and Adshead, J. (2016). Health resource use after robot-assisted surgery vs open and conventional laparoscopic techniques in oncology: analysis of english secondary care data for radical prostatectomy and partial nephrectomy. *BJU International*, 117(6):940–947. 14

Hung, A. J., Shah, S. H., Dalag, L., Shin, D., and Gill, I. S. (2015). Development and validation of a novel robotic procedure specific simulation platform: Partial nephrectomy. *Journal of Urology*, 194(2):520–526. 19

James, D. L. and Pai, D. K. (2003). Multiresolution green's function methods for interactive simulation of large-scale elastostatic objects. *ACM Transactions on Graphics*, 22(1):47–82. 22

Jing, M., Cui, Z., Fu, H., and Chen, X. (2021). Real-time deformation simulation of kidney surgery based on virtual reality. *Journal of Shanghai Jiaotong University (Science)*, 26(3):290–297. 38

Karimi, A. and Shojaei, A. (2017). Measurement of the mechanical properties of the human kidney. *IRBM*, 38(5):292–297. 28, 53

Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331. 25

Kerdok, A. E., Cotin, S. M., Ottensmeyer, M. P., Galea, A. M., Howe, R. D., and Dawson, S. L. (2003). Truth cube: Establishing physical standards for soft tissue simulation. *Medical Image Analysis*, 7(3):283–291. 25

Kim, J., Choi, C., De, S., and Srinivasan, M. A. (2007). Virtual surgery simulation for medical training using multi-resolution organ models. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 3(2):149–158. 22, 23

Kong, S.-H., Haouchine, N., Soares, R., Klymchenko, A., Andreiuk, B., Marques, B., Shabat, G., Piechaud, T., Diana, M., Cotin, S., and Marescaux, J. (2016). Robust augmented reality registration method for localization of solid organs' tumors using CT-derived virtual biomechanical model and fluorescent fiducials. *Springer*, 31(7):2863–2871. v, 18

L. da Frota Moreira, P., Peterlik, I., Herink, M., Duriez, C., Cotin, S., and Misra, S. (2013). Modelling prostate deformation: Sofa versus experiments. *Mechanical engineering research*, 3(2):64–72. eemcs-eprint-23873. 37

Lastrico, R. (2021). Attached github repository. 55

Lee, H., Nguyen, N. H., Hwang, S. I., Lee, H. J., Hong, S. K., and Byun, S.-S. (2018). Personalized 3d kidney model produced by rapid prototyping method and its usefulness in clinical applications. *International braz j urol*, 44(5):952–957. 29

Lerotic, M., Chung, A. J., Clark, J., Valibeik, S., and Yang, G.-Z. (2008). Dynamic view expansion for enhanced navigation in natural orifice transluminal endoscopic surgery. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2008*, pages 467–475. Springer Berlin Heidelberg. 16

Lerotic, M., Chung, A. J., Mylonas, G., and Yang, G.-Z. (2007). pq-space based non-photorealistic rendering for augmented reality. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2007*, pages 102–109. Springer Berlin Heidelberg. 16

Li, W. (2018). Biomechanical property and modelling of venous wall. *Progress in Biophysics and Molecular Biology*, 133:56–75. 28, 46, 49

Liu, Y., Kerdok, A. E., and Howe, R. D. (2004). A nonlinear finite element model of soft tissue indentation. In *Medical Simulation*, pages 67–76. Springer Berlin Heidelberg. 25

Long, J.-A., Yakoubi, R., Lee, B., Guillotreau, J., Autorino, R., Laydner, H., Eyraud, R., Stein, R. J., Kaouk, J. H., and Haber, G.-P. (2012). Robotic versus laparoscopic partial nephrectomy for complex tumors: Comparison of perioperative outcomes. *European Urology*, 61(6):1257–1262. 13

Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques - SIGGRAPH '87*. ACM Press. 29

Luboz, V., Promayon, E., Chagnon, G., Alonso, T., Favier, D., Barthod, C., and Payan, Y. (2012). Validation of a light aspiration device for in vivo soft tissue characterization (LASTIC). In *Studies in Mechanobiology, Tissue Engineering and Biomaterials*, pages 243–256. Springer Berlin Heidelberg. 28

Marchal, M., Promayon, E., and Troccaz, J. (2006). Simulating Prostate Surgical Procedures with a Discrete Soft Tissue Model. In Ed. C. Mendoza, I. N.,

editor, *Third Eurographics Workshop in Virtual Reality Interactions and Physical Simulation*, pages pp. 109–118, Madrid, Spain. EG Eurographics digital library. 38

Meier, U., López, O., Monserrat, C., Juan, M., and Alcañiz, M. (2005). Real-time deformable models for surgery simulation: a survey. *Computer Methods and Programs in Biomedicine*, 77(3):183–197. 20

Michiels, C., Jambon, E., and Bernhard, J. C. (2019). Measurement of the accuracy of 3d-printed medical models to be used for robot-assisted partial nephrectomy. *American Journal of Roentgenology*, 213(3):626–631. 29

Misra, S., Okamura, A. M., and Ramesh, K. T. (2007). Force feedback is noticeably different for linear versus nonlinear elastic tissue models. In *Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC'07)*, pages 519–524. 21

Misra, S., Ramesh, K. T., and Okamura, A. M. (2008). Modeling of tool-tissue interactions for computer-based surgical simulation: A literature review. *Presence: Teleoperators and Virtual Environments*, 17(5):463–491. 23

Mohareri, O., Ischia, J., Black, P. C., Schneider, C., Lobo, J., Goldenberg, L., and Salcudean, S. E. (2015). Intraoperative registered transrectal ultrasound guidance for robot-assisted laparoscopic radical prostatectomy. *Journal of Urology*, 193(1):302–312. 16

Montagnat, J. and Delingette, H. (1997). Volumetric medical images segmentation using shape constrained deformable models. In *Lecture Notes in Computer Science*, pages 13–22. Springer Berlin Heidelberg. 26, 29

Montagnat, J., Delingette, H., and Ayache, N. (2001). A review of deformable surfaces: topology, geometry and deformation. *Image and Vision Computing*, 19(14):1023–1040. 34

Mountney, P. and Yang, G.-Z. (2009). Dynamic view expansion for minimally invasive surgery using simultaneous localization and mapping. In *2009 Annual*

International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE. 16

Murphy, D., Challacombe, B., Khan, M. S., and Dasgupta, P. (2006). Robotic technology in urology. *Postgraduate Medical Journal*, 82(973):743–747. 9

Natali, A., Pavan, P., Carniel, E., Lucisano, M., and Taglialavoro, G. (2005). Anisotropic elasto-damage constitutive model for the biomechanical analysis of tendons. *Medical Engineering & Physics*, 27(3):209–214. 22

Okamura, A. M. (2009). Haptic feedback in robot-assisted minimally invasive surgery. *Current Opinion in Urology*, 19(1):102–107. 15

Ortmaier, T., Deml, B., Kubler, B., Passig, G., Reintsema, D., and Seibold, U. (2007). Robot assisted force feedback surgery. In *Springer Tracts in Advanced Robotics*, pages 361–379. Springer Berlin Heidelberg. 15

Ottensmeyer, M. P. and Salisbury, J. K. (2001). In vivo data acquisition instrument for solid organ mechanical property measurement. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2001*, pages 975–982. Springer Berlin Heidelberg. 28

Overtoom, E. M., Horeman, T., Jansen, F.-W., Dankelman, J., and Schreuder, H. W. R. (2019). Haptic feedback, force feedback, and force-sensing in simulation training for laparoscopy: A systematic overview. *Journal of Surgical Education*, 76(1):242–261. 17

Pacchierotti, C., Sinclair, S., Solazzi, M., Frisoli, A., Hayward, V., and Prattichizzo, D. (2017). Wearable haptic systems for the fingertip and the hand: Taxonomy, review, and perspectives. *IEEE Transactions on Haptics*, 10(4):580–600. 17

Pan, S. and Dawant, B. M. (2001). −title>automatic 3d segmentation of the liver from abdominal CT images: a level-set approach−/title>. In Sonka, M. and Hanson, K. M., editors, *Medical Imaging 2001: Image Processing*. SPIE. 29

Pearce, S. M., Golan, S., Gorin, M. A., Luckenbaugh, A. N., Williams, S. B., Ward, J. F., Montgomery, J. S., Hafez, K. S., Weizer, A. Z., Pierorazio, P. M., Allaf, M. E., and Eggener, S. E. (2017). Safety and early oncologic effectiveness of primary robotic retroperitoneal lymph node dissection for nonseminomatous germ cell testicular cancer. *European Urology*, 71(3):476–482. 6

Pelosi, G. (2007). The finite-element method, part i: R. l. courant [historical corner]. *IEEE Antennas and Propagation Magazine*, 49(2):180–182. 23

Picinbono, G., Delingette, H., and Ayache, N. (2000). Real-time large displacement elasticity for surgery simulation: Non-linear tensor-mass model. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2000*, pages 643–652. Springer Berlin Heidelberg. 25

Picinbono, G., Delingette, H., and Ayache, N. (2001). Nonlinear and anisotropic elastic soft tissue models for medical simulation. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*. IEEE. 21

Porpiglia, F., Amparore, D., Checcucci, E., Manfredi, M., Stura, I., Migliaretti, G., Autorino, R., Ficarra, V., and Fiori, C. (2019). Three-dimensional virtual imaging of renal tumours: a new tool to improve the accuracy of nephrometry scores. *BJU International*, 124(6):945–954. 29

Porpiglia, F., Checcucci, E., Amparore, D., Piramide, F., Volpi, G., Granato, S., Verri, P., Manfredi, M., Bellin, A., Piazzolla, P., Autorino, R., Morra, I., Fiori, C., and Mottrie, A. (2020). Three-dimensional augmented reality robot-assisted partial nephrectomy in case of complex tumours (PADUA >=10): A new intraoperative tool overcoming the ultrasound guidance. *European Urology*, 78(2):229–238. 13, 17

Porpiglia, F., Fiori, C., Checcucci, E., Amparore, D., and Bertolo, R. (2018). Hyperaccuracy three-dimensional reconstruction is able to maximize the efficacy of selective clamping during robot-assisted partial nephrectomy for complex renal masses. *European Urology*, 74(5):651–660. 29

Porpiglia, F., Morra, I., Chiarissi, M. L., Manfredi, M., Mele, F., Grande, S., Ragni, F., Poggio, M., and Fiori, C. (2013). Randomised controlled trial comparing laparoscopic and robot-assisted radical prostatectomy. *European Urology*, 63(4):606–614. 14

Porter, J. and Blau, E. (2020). Robotic-assisted partial nephrectomy. *Current Opinion in Urology*, 30(1):79–82. 13

Puso, M. A. and Weiss, J. A. (1998). Finite element implementation of anisotropic quasi-linear viscoelasticity using a discrete spectrum approximation. *Journal of Biomechanical Engineering*, 120(1):62–70. 25

Quraishi, M. K., Latif, E. R., Thomas, M., Eddy, B., Mazzone, E., and Mottrie, A. (2020). Robot-assisted partial nephrectomy: Evolving techniques. In *Evolving Trends in Kidney Cancer*. IntechOpen. 13

Reis, G., Yilmaz, M., Rambach, J., Pagani, A., Suarez-Ibarrola, R., Miernik, A., Lesur, P., and Minaskan, N. (2021). Mixed reality applications in urology: Requirements and future potential. *Annals of Medicine and Surgery*, 66:102394. 17

Ruder, S. (2017). An overview of gradient descent optimization algorithms. 71

Samur, E., Sedef, M., Basdogan, C., Avtan, L., and Duzgun, O. (2007). A robotic indenter for minimally invasive measurement and characterization of soft tissue response. *Medical Image Analysis*, 11(4):361–373. 25, 28

Schiavina, R., Bianchi, L., Borghesi, M., Chessa, F., Cercenelli, L., Marcelli, E., and Brunocilla, E. (2019). Three-dimensional digital reconstruction of renal model to guide preoperative planning of robot-assisted partial nephrectomy. *International Journal of Urology*, 26(9):931–932. 29

Schwenninger, D., Schumann, S., and Guttmann, J. (2011). In vivo characterization of mechanical tissue properties of internal organs using endoscopic microscopy and inverse finite element analysis. *Journal of Biomechanics*, 44(3):487–493. 28

Shirk, J. D., Thiel, D. D., Wallen, E. M., Linehan, J. M., White, W. M., Badani, K. K., and Porter, J. R. (2019). Effect of 3-dimensional virtual reality models for surgical planning of robotic-assisted partial nephrectomy on surgical outcomes. *JAMA Network Open*, 2(9):e1911598. 29

Stoyanov, D. and Yang, G.-Z. (2007). Stabilization of image motion for robotic assisted beating heart surgery. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2007*, pages 417–424. Springer Berlin Heidelberg. 16

Tan, A., Ashrafian, H., Scott, A. J., Mason, S. E., Harling, L., Athanasiou, T., and Darzi, A. (2016). Robotic surgery: disruptive innovation or unfulfilled promise? a systematic review and meta-analysis of the first 30 years. *Surgical Endoscopy*, 30(10):4330–4352. v, 6, 7

Tandogdu, Z., Vale, L., Fraser, C., and Ramsay, C. (2015). A systematic review of economic evaluations of the use of robotic assisted laparoscopy in surgery compared with open or laparoscopic surgery. *Applied Health Economics and Health Policy*, 13(5):457–467. 14

Tay, B., Stylopoulos, N., De, S., Rattner, D., and Srinivasan, M. (2002). Measurement of in-vivo force response of intra-abdominal soft tissues for surgical simulation. *Studies in health technology and informatics*, 85:514—519. 28

Villard, P.-F., Bourne, W., and Bello, F. (2008). Modelling organ deformation using mass-springs and tensional integrity. In Bello, F. and Edwards, P. J. E., editors, *Biomedical Simulation*, Berlin, Heidelberg. Springer Berlin Heidelberg. 26

Vito, R. P. and Dixon, S. A. (2003). Blood vessel constitutive models—1995–2002. *Annual Review of Biomedical Engineering*, 5(1):413–439. 21

Vittori, G. (2013). Open versus robotic-assisted partial nephrectomy: a multicenter comparison study of perioperative results and complications. *World Journal of Urology*, 32(1):287–293. 13

Vlachos, A., Peters, J., Boyd, C., and Mitchell, J. L. (2001). Curved PN triangles. In *Proceedings of the 2001 symposium on Interactive 3D graphics - SI3D '01*. ACM Press. 23

Wake, N., Rude, T., Kang, S. K., Stifelman, M. D., Borin, J. F., Sodickson, D. K., Huang, W. C., and Chandarana, H. (2017). 3d printed renal cancer models derived from MRI data: application in pre-surgical planning. *Abdominal Radiology*, 42(5):1501–1509. 29

Wolanski, P., Chabert, C., Jones, L., Mullavey, T., Walsh, S., and Gianduzzo, T. (2012). Preliminary results of robot-assisted laparoscopic radical prostatectomy (RALP) after fellowship training and experience in laparoscopic radical prostatectomy (LRP). *BJU International*, 110:64–70. 14

Wu, W. and Heng, P. A. (2005). An improved scheme of an interactive finite element model for 3d soft-tissue cutting and deformation. *The Visual Computer*, 21(8-10):707–716. 25

Wu, X., Downes, M. S., Goktekin, T., and Tendick, F. (2001). Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. *Computer Graphics Forum*, 20(3):349–358. 25

Xu, L., Lu, Y., and Liu, Q. (2018). Integrating viscoelastic mass spring dampers into position-based dynamics to simulate soft tissue deformation in real time. *Royal Society Open Science*, 5(2):171587. 26

Yaxley, J. W., Coughlin, G. D., Chambers, S. K., Occhipinti, S., Samaratunga, H., Zajdlewicz, L., Dunglison, N., Carter, R., Williams, S., Payton, D. J., Perry-Keene, J., Lavin, M. F., and Gardiner, R. A. (2016). Robot-assisted laparoscopic prostatectomy versus open radical retropubic prostatectomy: early outcomes from a randomised controlled phase 3 study. *The Lancet*, 388(10049):1057–1066. 29

yun Yao, H., Hayward, V., and Ellis, R. E. (2005). A tactile enhancement instru-

ment for minimally invasive surgery. *Computer Aided Surgery*, 10(4):233–239. 17

Zhang, X., Shen, Z., Zhong, S., Zhu, Z., Wang, X., and Xu, T. (2013). Comparison of peri-operative outcomes of robot-assisted vs laparoscopic partial nephrectomy: a meta-analysis. *BJU International*, 112(8):1133–1142. 13

Zhu, B., Gu, L., Zhang, J., Yan, Z., Pan, L., and Zhao, Q. (2008). Simulation of organ deformation using boundary element method and meshless shape matching. In *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE. 27