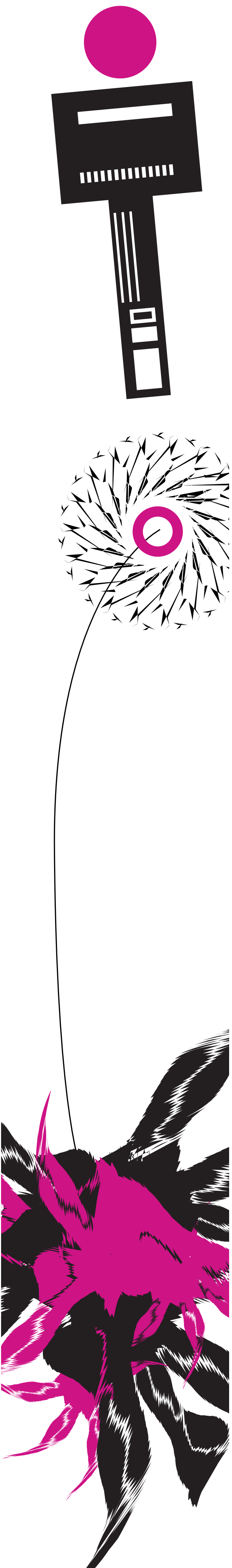

RAM ROBOTICS AND MECHATRONICS

MODELLING AND SHARED CONTROL OF ENDOVASCULAR SURGICAL INSTRUMENTS IN SOFA

J. (Jochem) Bos

MSC ASSIGNMENT

May, 2023

015RaM2023
Robotics and Mechatronics
EEMathCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Preface

This thesis was written for the completion of my MSc in Systems & Control at the University of Twente's Robotics and Mechatronics group. I am very grateful for the support and guidance provided by doctors Dagnino and Kundrat throughout the research process. Their kind and insightful supervision, as well as their remarkable patience, has been instrumental in the completion of this project.

Abstract

Cardiovascular disease is one of the primary causes of the death in the developed world. A significant subset of these diseases is treated with endovascular surgery using a catheter and guidewire combination. The manual insertion of surgical instruments into the veins of patients carries risks towards the structural integrity of the blood vessels, and therefore the health of the patient. This thesis describes a hardware-in-the-loop framework including a robotic control platform in the SOFA simulation environment, connected and controlled via Python. The developed framework includes instrument force assessment, Bézier spline-based trajectory planning for endovascular instruments and an implementation of shared attitude control. It provides an interactive simulation with the aim to test a control implementationAnonymization that might prevent dangerous situations from occurring via a combined approach of haptic feedback, input-blending shared control and visual guidance. Furthermore, the thesis presents the base for a surgical training network in simulation that can offer real-time visual and haptic feedback during the simulated procedure. Framework and control performance were evaluated qualitatively and quantitatively on different operators (N=3) during two to four separate trials with four different levels of haptic feedback and guidance. From the final set of trials is concluded that the implemented control method serves at a guidance level of 0.25 is able to reduce the mean navigational error by 17.4% ($p < 0.001$), and that haptic feedback and shared control at the low (0.25) and medium levels (0.66) can reduce the mean tip force by 14.6% and 22.6% respectively ($p < 0.001$). During qualitative evaluation of framework performance, it is concluded that the core framework performs as intended and forms a solid base to expand upon for both control development and surgical training purposes.

Contents

Chapter 1

Introduction

1.1 Background

Endovascular surgery, operating within the blood vessels of the patient, is a minimally invasive technique used to combat many types of vascular disease. Typically, it is performed as part of the discipline of interventional radiology, which combines endovascular tools with imaging methods such as x-ray fluoroscopy or magnetic resonance imaging (MRI). The specialized set of tools available to perform these operations consists of a catheter and guidewire, as well as other supporting tools. The guidewire is used by the surgeon to navigate the vessels of the patient and get the catheter, which slides over the guidewire, in position. Once positioned, the catheter can fulfill many functions for the surgeon such as the administration of fluids and providing access for other surgical instruments, as well as many others. The navigation of these tools with often limited visual feedback is a heavily trained and slowly acquired skill [1] and while patient injuries are limited in amount, they can be severe. Even when no complications occur, a reduction in forces administered to the patient can reduce recovery times and the necessary dose of anaesthesia [2]. Additionally, any device that can remove the operator from the radiation sources necessary for X-Ray fluoroscopy will reduce the exposure for the surgeon.

This research was carried out at the Robotics and Mechatronics (RaM) group at the University of Twente. The project flows from the work done by Dr. Dagnino and Dr.-ing Kundrat in the previous years in designing an MR-safe catheterization robot [3]. While other similar devices exist [4], the distinguishing feature of this robot is its ability to operate during an MRI-scan as well as in the more traditional setting with visual feedback from X-ray fluoroscopy. The robot has been the product of years of research, and at this point has been successfully tested in-vivo in an animal setting [5]. Now that the platform is shaping up for the transition towards clinical validation, it is the perfect place to start research into how the most benefit can be extracted from it for the surgeon that operates it and how simulation, guidance and control can be valuable tools for this objective.

The automation of surgical procedures has been a well-discussed topic in recent years [6]. There are many potential benefits to automating the work in the operating room: potential reduction of dangerous errors, safer training, shorter recovery times due to increased precision, reduced radiation exposure to the surgeon and many others. It is therefore enticing to want to automate the whole process from beginning to end, but it is not always clear this is the best solution. Regardless of approach, the dynamics of the instruments are complex and under-actuated and as such heavily dependent on the environment. Anytime a corner needs to be passed by the surgeon, normal forces from the walls of the vessels are necessary to steer the instrument at all. The instruments have a natural curve that is designed for a specific section of the human vascular anatomy. Control of this curvature during the operation is limited, and can only be done by the interaction between the degrees of stiffness of the two instruments by sliding and retracting the catheter over the guidewire, or by bending the instruments due to collisions with the vessel walls.

These dynamics make that it is not very feasible to do full motion planning or complete analytic control on the instruments. This has led researchers into exploring machine-learning approaches, but their performance is often limited in scope and hard to generalize [7] [8]. While great steps have been made towards performing tasks with reinforcement learning methods such as Deep Q-Networks and Deep Deterministic Policy Gradient networks [9], these suffer from relatively low success rates compared to human operators as well as the necessity to pre-train the model on a specific anatomy case. Patient anatomies are varied and the is a large variation in the structure of the same portion of anatomy between cases, and this has formed a fundamental roadblock so far. Some researchers have had success within a limited scope, such as in 2D environments with discrete action spaces by learning from demonstrations with a Deep Q Network [10]. As far as known to the author, no serious progress has been made towards implementing fully autonomous navigation that will feasibly conform to the high performance and safety requirements in a surgical environment. This is in part causes by the difficulty in bridging the gap between simulation and reality. While some researchers have had success in porting their model to a real environment, this has inevitably caused a drop in success rates even when similarity between the actual anatomy and used model are high [11].

Many of the authors cited above have the aim of improving navigation success rates and completion times. While this is a well-defined and interesting metric from a technical perspective, completion times and navigational success rates are not actually primary metrics of performance in endovascular surgeons [12]: safety and precision are of greater value. Beside the issues named above, automation also makes little to no use of one of the most valuable resources in interventional medicine: the expertise and experience of the surgeons themselves. A way to combine this expertise with part of the potential safety improvements of automation can have many benefits, not least of which being that it could seriously improve the currently hesitant acceptance of automated tools by the surgeons themselves.

1.2 Goal of the assignment

This document presents a method for shared control of endovascular surgery instrument, catheters and guidewires, using a master-slave hardware setup that is designed to preserve the expertise of the operating surgeon but assist where mistakes are made. This method combines several parts: the hardware designed and built for [3], finite element modeling of the instrument, spline-based trajectory planning methods, attitude control and guided operation for the other degrees of freedom. As such, not just the pathing method and control designs but the framework that was built around it are the contribution demonstrated by this work.

The aim of the framework is to provide the ground work as a future testing ground for endovascular instrument control and surgical training, and to demonstrate the shared control method presented in this document. It includes a complete pipeline for the generation of vascular environments for simulation, simulated dynamics of the instruments within these environments and implemented CAN-communication with the external master. Experiments were conducted to assess both the quantitative and qualitative performance of the simulation and framework in general. In the discussion, the merits and shortcomings of the current implementation are discussed as well as directions for future research and improvement.

The objective for the implemented pathing and control methods is to reduce the amount and severity of forces expressed by the tip on the vessel walls via provided feedback to the user and direct intervention, as well as assist in the navigation towards a pre-set trajectory point. Navigational success is assessed via mean error in regard to the common path. Additionally, computational performance of the framework, trial completion times and collision counts are investigated.

1.3 Report organization

The report is organized in six main chapters, excluding introductions and discussion. In chapter 2, *literature*, the theoretical background to the thesis is discussed. This chapter touches on the working principles of SOFA and specifically constraint handling therein, trajectory planning using breadth-first search and Bézier-splines, and a literary overview shared control methods. In chapter 3, *system design*, the primary components of the system are discussed, focusing on the hardware-in-the-loop implementation and communication pipeline, as well as the main software implementation of the framework. Chapter 4, *modelling*, describes the modelling choices for the implementation of the surgical instruments and the simulated environment in SOFA, as well as the steps taken to optimize the simulation. Next, in chapter 5, *pathing and shared control framework*, the designs and implementation of the trajectory planning methods through the virtual anatomy, and the shared control implementation guiding the operator are introduced. Chapter 6, *evaluation*, describes the testing of the framework and control methods via user trials, and finally chapter 7, *discussion*, comments on the findings during these trials and the benefits and suggests possible improvements of the presented platform, presents research outcomes and provides recommendations for the future. The main body of the work ends with a conclusion, followed by appendices and bibliography.

Chapter 2

Literature

2.1 Simulation in SOFA

The Simulation Open Framework Architecture (SOFA) [13] was first publicly launched in 2007 by a group of French scientists and engineers at the French National Institute for Research in Digital Science and Technology (INRIA) [13]. The framework was and is serviceable for a broad range of applications, but from the beginning one of the primary foci of the framework was medical simulation at interactive frame rates.

The framework is written almost entirely in C++ and can be interacted with via scenes. These were originally written in XML but can now be scripted in Python via the SOFAPython API, which is the medium used for this piece of research. A thorough explanation of the workings of SOFA is beyond the scope of this document, but there are a few key points that are relevant. SOFA is primarily a FEM-based simulator and allows for multiple representations of the same object for different dimensions of the simulation. A simulated blood vessel can have a coarse tetrahedral mechanical model, a finer triangle-based collision model and an even finer visual model. These models and their properties are stored in *nodes* that contains one specific representation. Such a node contains *components* that may be meshes, containers for degrees of freedom, force fields for interactions, and numerical solvers. These nodes are stored and visualised in a tree-like structure called the *scene graph*, and the simulation is solved in each time step by the solvers visiting every node in the tree in a structured manner.

A lot of flexibility is provided by the modular structure of the simulator. This flexibility has allowed many creators to design plugins for SOFA that add new components or open up whole new fields for application within it. One such plugin, the Beam Adapter plugin [14], is of significant importance for this thesis, as it models the mechanical properties of endovascular instruments. A list of other dependencies for this work can be found in Appendix B

2.1.1 Catheter and guidewire modelling in SOFA

The Beam Adapter plugin is based on Kirchhoff rod theory (For background, see: [15]) and Cosserats further refinement of this theory [16]. The work of primary importance for the implementation of Kirchhoff's and Cosserat's work into SOFA is done by C. Duriez [17], by first introducing the Beam Adapter plugin. This implementation uses a 12 x 12 sparse symmetric elementary stiffness matrix, $\boldsymbol{K}_E$, for each beam segment to relate the positions and rotations to the applied forces and torques at the ends, under the assumption that the deformations in the beam are small relative to the deformation of the total structure. This stiffness matrix is originally computed as $\bar{\boldsymbol{K}}_E$ in local coordinates where the origin of the frame is the initial position of the beam base:

$$\boldsymbol{f}_e = \bar{\boldsymbol{K}}_E(\bar{\boldsymbol{u}} - \bar{\boldsymbol{u}}_{\boldsymbol{0}}) = \bar{\boldsymbol{K}}_E \delta \bar{\boldsymbol{u}} \tag{2.1}$$

where $\boldsymbol{f}_e \in \mathbb{R}^{12\times1}$ is the column vector of external forces on the beam, $\bar{\boldsymbol{u}} \in \mathbb{R}^{12\times1}$ is the final configuration vector in the local frame, and $\bar{\boldsymbol{u}}_0$ is the initial configuration in the same frame.

To get the final configuration in the global reference frame, transformation matrix $\boldsymbol{\Lambda}(\boldsymbol{q})$ is applied according to eq. 2.2

$$\delta \bar{\boldsymbol{u}} = \boldsymbol{\Lambda}(\boldsymbol{q})\delta \boldsymbol{q} \tag{2.2}$$

The use of the stiffness matrix to solve for these kinds of problems is a fairly common approach, but the trick lies in the handling of the inherent non-linearities in the system. This approach handles this by not using the global displacement, but by defining a new reference state at every timestep, and thus re-calculating both the coordinate transformation and linearized displacement. The consequence of this however, is that elastic behaviour will only return to the last reference state, or the previous timestep. This is then solved by computing this elastic force separately before solving the system and adding it to the external forces in the global frame:

$$\boldsymbol{K}_E \boldsymbol{u} = \boldsymbol{f}_{ext} + \boldsymbol{f}_{el} \tag{2.3}$$

$$\boldsymbol{f}_{el} = -\boldsymbol{\Lambda}\gamma \bar{\boldsymbol{K}}_E(\boldsymbol{u} - \boldsymbol{u}_0) \tag{2.4}$$

where $\boldsymbol{f}_{ext}$ and $\boldsymbol{f}_{el} \in \mathbb{R}^{1\times 12}$ are the external and elastic force vectors and γ is a scalar damping coefficient. Bars indicate expression in the local frame.

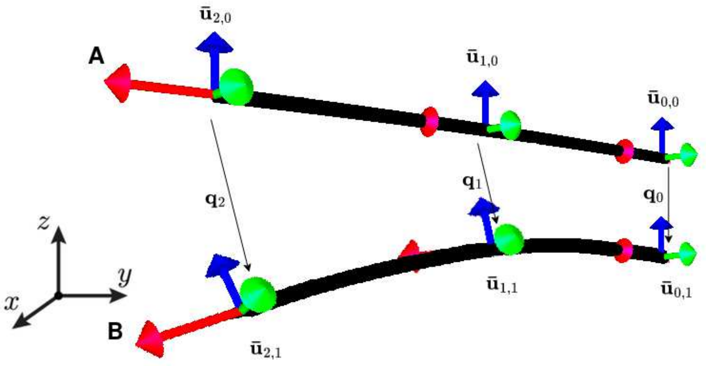

Figure 2.1: A wire-like structure consisting of beam elements in its initial configuration (A) and second configuration (B). $\boldsymbol{q}$ is the displacement in the global frame given by the axes in the bottom left. This figure displays the change for a single timestep, where each local frame $\bar{u}_{i,0}$ undergoes a linear transformation towards $\bar{u}_{i,1}$

Because the goal is to use this as a real-time tool, *matrix substructure analysis* [18] is applied to the full structure of the instrument. The essence of this technique is that in a larger structure such as the modeled catheter can be divided in smaller substructures such as the beam elemnents described above, that are constrained by each other at interface points. At the interface point, combining the sets of linearized equations for the substructures allows for the application of standard linear solvers to complex non-linear problems, in our case, a block term decomposition (BTD) solver [19]. In order, the resolution steps are boundary fixation, where the boundary between the system and the real world is treated as rigid, boundary relaxation, where internal forces are propagated to the boundary node, and flexibility assembling, where the two are put together to complete the computation of the unconstrained dynamics of the beam structure. While the BTD-solver is an efficient solver for large amounts of low-ranked tensors [19] such as in this case, the real computational effort comes in when the beam structure needs to move within a constrained space, such as the vascular mesh in this work.

2.1.2 Contacts and constraints

The theory behind handling contacts and constraints in SOFA is relevant, as the movement of a catheter-guidewire cannot be expressed by a set of equations independent from the environment: the instruments require friction and constraints to move in any other direction than the base-controllable degrees of freedom: advancement (depth) and axial rotation. All other degrees of freedom can only controlled through interaction: either between the two instruments or with the environment. Because deformation of the wall material is outside the scope of this research, we consider the vessel walls rigid.

SOFA handles constraints using the method of *Lagrange multipliers* (For background, see: [20]). The handling of these constraints is a computationally complex tasks; a naive computation would require tracking all points relative to all other points would scale quadratically with the amount of points. Instead, the algorithm is divided in several sub-steps that optimize the number of constraints that are considered as well as the resolution of these constraints. A diagram portraying the steps is shown in fig. 2.2

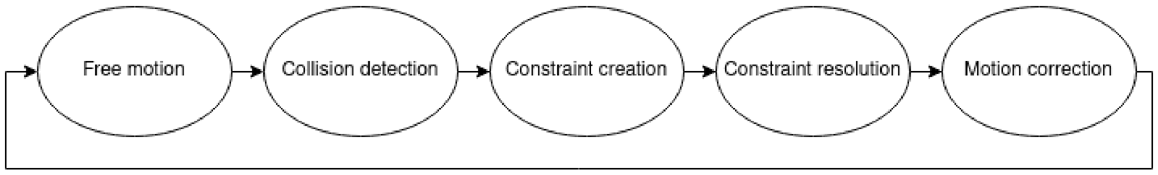

Figure 2.2: Diagram showing the constraint resolution cycle in SOFA. During the free motion step, the objects move uninhibited and may interpenetrate. Collisions are detected based on the SOFA collision detection algorithm [13]. After this, constraints are created and resolved using the Gauss-Seidel algorithm in constraint space. Finally, the initial free motion is corrected with the resolution of the constraints.

During the free motion steps, objects move unconstrained such as described for the instrument in the previous section. After this motion is computed, the collision detection algorithm is launched. This consists of a broad phase, a narrow phase and finally an intersection method. During the broad phase, a brute force approach is used to check if the axis-aligned bounding boxes of the objects are in collision with eachother. In the narrow phase, a bounding volume hierarchy (BVH) approach is used for each pair of collision models that is observed by the broad phase to be in collision. In this phase, every element of the two objects is given a bounding volume, and the overlap is checked between every pair. This part of the operation can be time consuming, especially for complex collision meshes. Pairs that intersect are passed to the intersection method. In our case, the intersection method is a cone projection method called *LocalMinDistance*, which projects cones from every contact point and calculates the minimal distance between the actual finite elements (not the bounding boxes).

When these steps are finished, constraint resolution can be initiated. This is done by formulating the constraint problem as in 2.6.

$$(\boldsymbol{M} + h\frac{\delta \boldsymbol{F}}{\delta \boldsymbol{v}} + h^2\frac{\delta \boldsymbol{F}}{\delta \boldsymbol{q}})d\boldsymbol{v} = -h^2\frac{\delta \boldsymbol{F}}{\delta \boldsymbol{q}}\boldsymbol{v}_i - h(\boldsymbol{f}_i + \boldsymbol{p}_f) + h\boldsymbol{H}^T\lambda \tag{2.5}$$

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} + h\boldsymbol{H}^T\lambda \tag{2.6}$$

The first equation is the explicit constraint problem. Here, $\boldsymbol{M}$ is the inertia matrix, h is the time interval, $\boldsymbol{F}$ is the matrix of internal forces, $\boldsymbol{v}$ is the velocity vector, $\boldsymbol{f_i}$ is the value of $\boldsymbol{F}$ at the beginning of timestep *h*, while $\boldsymbol{p_f}$ is the value of $\boldsymbol{P}(t)$, the matrix of external forces, at the end. The term $\boldsymbol{H}^T\lambda$ is the vector of constraint forces where $\boldsymbol{H}$ is the matrix of constraint directions and λ is the Lagrange-multiplier that must be solved for via numerical integration. The second equation in 2.6 represents the simplified non-linear system with system matrix $\boldsymbol{A}$. The only unknowns in the system are in the term $\boldsymbol{H}^T\lambda$.

The resolution of the linearized system is obtained via the Gauss-Seidel algorithm (See [21] [22]), an iterative method. The algorithm converges for positive definite and diagonally dominant systems, which is the case here as long as long as the internal forces are within reasonable limits. This approach has a computational complexity of $\mathcal{O}(M + N)$ for *M* contacts and *N* total nodes [22]. Unfortunately, the inter-dependency of constraints in the problem combined with the nature of the Gauss-Seidel algorithm result in a very long critical path, i.e. the amount of computational work that has to be done sequentially. This leads to a very bad fit for parallel computation and thus limitations to hardware scaling with increased model complexity.

2.2 Autonomy and control in vascular surgery

2.2.1 Autonomous navigation for vascular instruments

A lot has been written about robotic motion planning, especially those on wheels or wings. Motion planning for irregularly shaped robots and structures such as catheters is a scarcer field, although some work has been done in this regard. Two general approaches exist on a spectrum: machine-learning (ML) based planning and kinematics-based analytic planning. Below, a selection of works is discussed that at least fulfill the following conditions:

- High-level motion planning is implemented for either only a guidewire or both catheter and guidewire, either with or without low-level control implementation.
- The work assumes or proposes no additional degrees of freedom to the translation and axial rotation of the traditional catheter-guidewire combination. I.e.: the methods apply to standard endovascular instruments.

One such machine-learning-based paper is [23], where the authors attempt to reduce contact force intensities and durations via the application of a reinforcement learning network based on dynamic movement primitives. The network learns from human demonstration and acts in a aortic phantom. The authors find that their approach reduces mean and maximum forces by approximately 60% and 30% respectively, but increases duration of the operation. In [24], the authors propose a generative adversarial network composed of convolutional neural networks (CNN) and long short-term memory networks (LSTM). A method is proposed where the networks must reproduce the motions of an expert surgeon, to thereafter perform them autonomously. The authors conclude that while trajectory completion rates are increased compared to comparable methods, there is room for improvement when compared to expert manual performance. Another reinforcement learning approach is proposed in [10], who propose a Deep-Q reinforcement learning framework for full navigational autonomy of a guidewire tested in 2D and 3D phantoms. The authors show to be able to significantly reduce the amount of movements by specifically rewarding the model for conservativeness in this regard, as well as a 100% completion rate for the fully trained version. Lastly, in [25], an attempt is made to improve generalisation by applying a deep deterministic policy gradient (DDPG) approach to reinforcement learning on a 2D phantom. The DDPG-based high-level approach is combined with a neural network-controlled low level controller. While results in terms of success rates are promising, the method is only applied to 2D and generalisation is only demonstrated in different objectives within the same vascular phantom.

Analytic approaches have also been proposed to solve the motion planning problem. In [26], a two-phase approach is proposed for constrained motion planning of flexible surgical tools. The first phase consists of a breadth-first search algorithm that computes a centerline trajectory through the simplified vascular mesh. The second phase consists of a genetic algorithm that locally optimizes the path according to curvature metrics, guaranteeing the maximum amount of bending is not exceeded for the instrument. The authors conclude that this algorithm is specifically valuable in minimizing this curvature, which can be the cause of dangerous elastic forces due to instrument coiling, as well as somewhat improving operating time and total distance traversed. Unfortunately, the work does not include motion testing in simulation or phantom. In [27], a pre-operation planning pipeline stretching from surgical image segmentation to trajectory planning is proposed. The trajectory planning here is based on RRT combined with sequential convex optimization to satisfy curvature and path length constraints. In an older work [28], a fully analytical method based on guidewire tip kinematics is implemented which is successfully able to navigate to goal positions in swine autonomously via inverse-kinematic path planning combined with locally controlled obstacle evasion.

The non-exhaustive list of related work above demonstrates that the approaches are as diverse as the number of papers. There are many dimensions to consider in catheter navigation, but broadly the works above can be organized along a few axes that are of specific relevance to how the motion of the instrument is constructed:

- *full instrument autonomy* vs. *guidance*
- *machine-learning-based* vs. *kinematic methods*
- *trajectory planning* vs. *path planning*
- *safety-oriented* vs. *navigation-oriented*

The first axis, *full instrument autonomy* versus *guidance*, describes the degree to which researchers want to create a system is able to fully autonomously navigate the instrument. The other side, guidance, indicates that the surgeon should still be fully in the loop and only be guided by an algorithm. Autonomy always means that the ultimate aim is for the surgeon only to have a supervisory role, while guidance indicates that the surgeon still makes all mid to high-level decisions in the operation (e.g. taking a specific corner or entering with a specific instrument attitude). The axis *trajectory planning* versus *path planning* describes the order of the planned motion. Trajectory planning only defines a positional connection between beginning and end points, whereas path planning includes further derivatives such as velocity and acceleration. The last axis, *safety-oriented* versus *navigation-oriented*, describes whether the researchers aim for decreased occurrence of dangerous forces (safety-oriented) or better completion rates and times for the navigation (navigation-oriented). Table 2.1 displays how the above works can be placed. Note that the classification is not necessarily discrete, and as such the work is placed where it is closest to. This thesis will focus on safety-oriented trajectory planning applying guidance to an independent operator via analytic methods.

Work	Guid.-Aut.^{1}	ML-Kin.^{2}	Traj.-Path^3	Safety-Nav.^{4}
[23]	Autonomy	ML	Path	Safety *and* navigation*
[24]	Autonomy	ML	Trajectory	Navigation
[10]	Autonomy	ML	Path	Navigation
[25]	Autonomy	ML	Trajectory	Navigation
[26]	Guidance	Kinematic	Path	Safety *and* **navigation***
[27]	Autonomy	Kinematic	Trajectory	Navigation
[28]	Autonomy	Kinematic	Trajectory	Navigation
This work	Guidance	Kinematic	Trajectory	**Safety** *and* navigation*

Table 2.1: Categorization of papers regarding endovascular surgical instrument navigation. ^{1}: Guidance versus autonomy, ^{2}: machine-learning versus kinematics, ^{3}: trajectory planning versus path planning, ^{4}: safety-oriented versus navigation-oriented. $*$: some works focus on both safety and navigation. If there is an emphasis on either of the categories, this is in **bold**

2.2.2 Shared control

Of specific importance to the subject of this thesis is the matter of autonomy in surgical instruments. While, as listed in table 2.1, many researches have aimed to fully autonomize surgical instruments, there is an open question on whether this is the right approach. Clearly, there is a benefit in reducing the potential for vibrations and human error to carry through in the operation of a surgical instrument. Robotic surgery in many different operations has been found to be potentially safer and more consistent than manually operated surgery [29], but this still assumes the human does the decision-making. Autonomy is a step further, and is defined by minimal intervention or even supervision by a human operator. While some successful navigation autonomous navigation techniques have emerged, it is not clear whether they are general, robust and safe. Specifically in machine-learning-based approaches [23] [24] [10] [25], the "black box"-nature of the method prevents researchers from thoroughly understanding the operation of an algorithm. The lack of interpretability of these models also might stand in the way of the certification necessary for operation on human patients. Generalization and sample efficiency are problems that also apparent in these approaches, with the amount of training ranging from 1.000 episodes for a RL-agent on a single 2D anatomical phantom [10] to 350.000 frames of video for image-based pathing [24]. Not only ML-based approaches suffer from issues when trying to fully autonomize the surgical operation, kinematic and other analytical methods have trouble with this as well. Here, the problem lies more in the complexity of the problem from a mathematical or control perspective: the catheter and guidewire move in a chaotic environment where environmental interaction is key to the navigational success. Attempts to achieve full autonomy via a kinematic approach either only propose such a system without follow-up on implementation [27], or achieve a simplified system that can only perform sub-steps of the procedure autonomously [28].

Besides doubts about the current safety and efficacy of autonomous platforms in endovascular surgery, there are also practical, ethical and social questions bound to it. Acceptance of the idea of autonomous surgery among the general population in Europe, the United States, India and Brazil is low, and opinions about responsibility for errors are divided [30]. It is also not clear at all the professionals in the field would welcome reduced control over their operating rooms or carrying the partial responsibility over an autonomous system, although quantitative surveys in this regard have not been found. In surveys within the field of endovascular surgery [6] where the advancement of robotic devices is investigated, no mention is made, either direct or indirect, of a need for fully autonomous navigation. Instead, a need is stated for safer, more reliable robotic devices with haptic feedback, compatibility with most-used imaging techniques and a low entry barrier both for endovascular surgical training and daily use in the operating room.

Instead of aiming to fully autonomize the surgical instruments themselves, a better approach might be not to replace the skillful and well-trained surgeons that are already performing endovascular surgery, but to supplement these skills with a guided approach. A tentative step in this directions is already made in [26], but this work primarily investigates the pathing of the instrument, not the operation. For other tasks, such as suturing, shared control that shifts between the surgeon and an autonomous system has already been implemented by steering the surgeon via haptic feedback towards an optimal needle angle [31]. A similar approach with a surgical cutting tool with haptic guidance is implemented in [32]. Specific to robotic catheter and guidewire control, however, no such publication focusing on the aspect of shared control has been found. More generally, the blending of robot and operator input is described in [33], which describes several approaches to the *policy blending* necessary to combine the robotic and manual inputs. The authors define levels of prediction and arbitration, where prediction describes if the shared autonomy policy tries to predict future user behaviour, and the arbitration describes how the fusion of inputs is defined. As prediction naturally relies on the predictability of the kinematics, the proposed method in this thesis primarily deals with the arbitration dimension of policy blending.

2.3 Bézier splines

Because of the long and flexible nature of the instruments, it cannot change direction suddenly. Constraints are necessary on the smoothness and curvature of the path. For this reason, Bézier curves are heavily used in the proposed pathing algorithm, as they have been in other motion planning applications [34]. Bézier curves are constructed using control points using a parametric function. Given a set of points $(p_0, p_1, .., p_n)$ a Bézier curve of the n-th order can be obtained for all points $t \in [0, 1]$ via:

$$B(n,t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i p_i w_i \tag{2.7}$$

Where w_i is a weight controlling how strongly a specific control point "pulls" on the curve. As every axis is defined independently, this holds for 1, 2 or 3 dimensions.

Because the description of the curve is a sum of polynomials, Bézier curves can be neatly expressed in matrix form as well, as is shown below for n=3:

$$\boldsymbol{B}(t) = \boldsymbol{tMs} \tag{2.8}$$

$$\boldsymbol{B}(t) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} \tag{2.9}$$

Where M has the useful properties that is both invertible and triangular [34]

When Bézier curves are fitted together, they form a Bézier spline. As fitting a single curve through a long path is very computationally expensive, it is much more feasibly to path subsections and then fit them together according to a set of constraints. These constraints relate to the continuity in derivatives of the connected curves; C^0 continuity just means there are no gaps between the curves, C^1 indicates continuity between first derivatives, C^2 between second derivatives and so forth. As such, a C^2-continuous path has no jumps in acceleration along the path. Given a set of control points, the derivative of a given Bézier curve is [35]:

$$B'(n,t) = \sum_{i=0}^{k} \binom{k}{i} (1-t)^{k-i} t^i (w_{i+1} - w_i) \tag{2.10}$$

Where $k = n - 1$. This shows that the derivative of a Bézier curve of order n is in itself a Bézier curve of order n-1, with new weights. These new weights in $\boldsymbol{w}'$ are related to the old weights $\boldsymbol{w}$:

$$w'_i = (n-1) * (w_{i+1} - w_i) \tag{2.11}$$

Because of this property, we can guarantee C^2-continuity for two 3rd order Béziers with control points (weights) p_i^1 and p_i^2 ($i \in [0, 3]$), respectively, by satisfying the following constraints [36]:

$$p_3^1 - 2p_2^1 + 2p_1^1 = p_2^2 - 2p_1^2 + p_0^2 \tag{2.12}$$

Figure 2.3 shows how Bézier curves can be combined into a spline and how the manipulation of control points can achieve C^2-continuity.

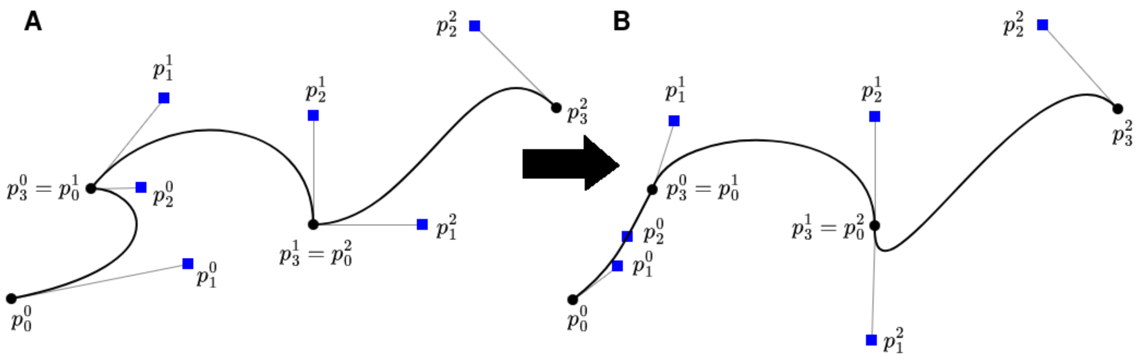

Figure 2.3: **A**. Diagram portraying the construction of a Bézier spline from multiple Bézier curves. The curve is not C^2-continuous. **B**. Rearrangement of the control points provides a (non-unique) C^2-continuous spline

Chapter 3

System design

This chapter provides an overview of the framework as implemented. The aim is to show the interaction between hardware and software components, with more detail about modelling, pathing and control specifics in the subsequent chapters. Integration and communication with the master device as well as global software architecture are highlighted, showing the interaction between various components and providing a birds-eye view of the framework.

3.1 System overview

Fig. 3.1 shows the primary interacting parts of the system in both hardware and software domains. The system consists roughly of three components: the master device, python framework and SOFA scene. It follows a control hierarchy where motion commands from the master are fed to the python framework where they are processed and compensation is added before they are finally executed in the SOFA scene.

As the master device is an existing project not specifically or exclusively designed for work, the reader is referred to [3] for a more detailed outline of the inner workings and technical design. For this reason, this document will only touch on the basics of operation for this hardware in section 3.2. Additionally, the slave device the master is connected to for physical operation is omitted in the diagram, as it is not part of the hardware-in-the-loop system as presented. Instead of the physical slave device, the operation is simulated in SOFA, the rightmost block in fig. 3.1. This is done in a SOFA scene consisting of several nodes that contain sub-nodes, components or functions. The full tree contains around 80 nodes over 6 children (branches), so it is displayed in a simplified form. Interaction with the SOFA scene is done

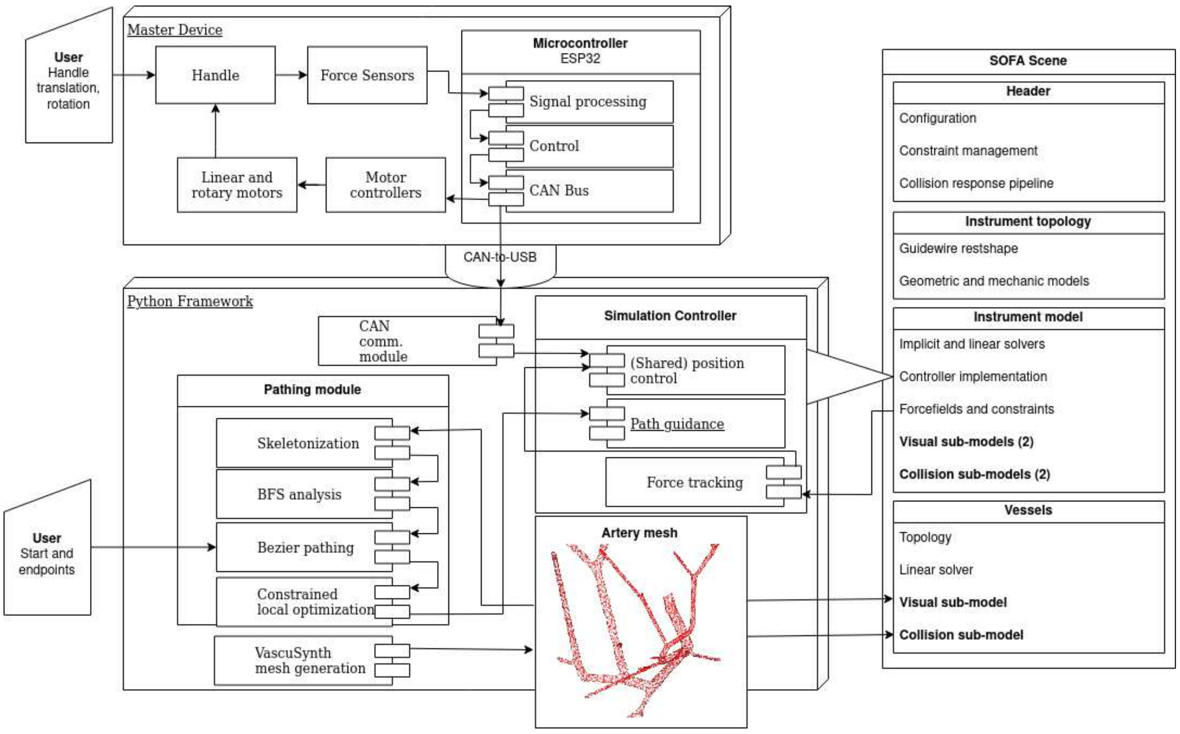

Figure 3.1: Overview of the complete system. The figure shows the three main components, the master device, python framework and SOFA scene and their sub-components. Operational flow of commands goes from the master device through the python framework to the SOFA scene, but there is an exchange of data and haptic feedback commands in the other direction as well.

via the Python framework in the bottom left of the figure. This Python framework defines the SOFA scene beforehand, and can control the scene during runtime via the simulation controller. This simulation controller is called at every time step to handle communication queues, I/O operations, control calculations, and visual and haptic feedback. The simulation controller thus serves as the brain of the system, directing all other components. Besides the simulation controller, the framework contains modules for pathing and CAN communication, as well as integration with the VascuSynth [37] mesh generator.

3.2 Hardware

The master device used for the hardware-in-the-loop simulation framework is displayed in fig. 3.2 below. The device is operated via the printed handle on a slotted shaft. This handle can be shifted and/or rotated to emulate the operations done by a endovascular surgeon in a manual catheterization set-up. The operation of the master device was specifically designed to require similar input motions to those in the operating room, to facilitate an easy transfer for surgical professionals and lower the barrier of entry.

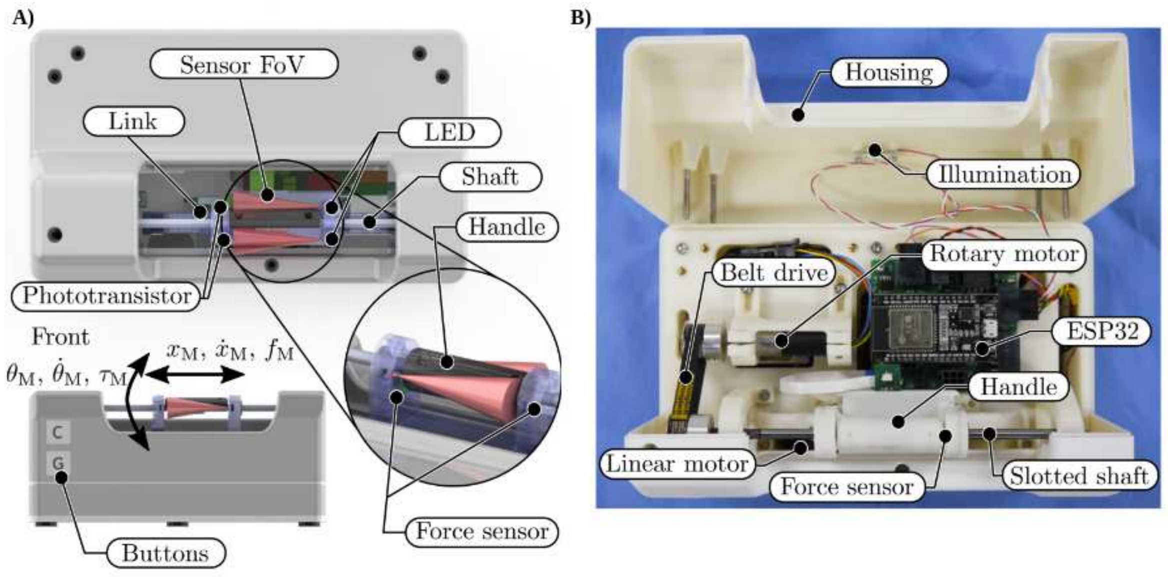

Figure 3.2: A) Top view of the covered master device and detail of handle.
B) Opened top view of the master device, showing primary electronic components. Adapted from Kundrat, Dagnino et al. (2021, [3])

To move the handle in either the translational or rotational direction, the phototransistors in the handle must be sufficiently covered for the device to register it as gripped. Force (f_M), torque (τ_M) and displacements sensors pick up the relevant linear ($x_M, \dot{x_M}$) and angular ($\theta_M, \dot{\theta_M}$) states, which is combined with the grip state (ρ_M), selected instrument (catheter or guidewire, λ_M) and intention state (feeding or retracting, κ_M) into a state vector q_M. The selected instrument is provided by the user via the "G" and "C" buttons on the side of the device. The intention state is a derived state, but is not used in the context of the rest of this work. Use of the device requires a calibration sequence beforehand. This sequence consists of powering the device, moving the handle back and forth and sending an acknowledgement request from the receiving interface; in this case the PC running the framework. Upon receiving acknowledgement, the handle will move from end to end and will eventually home in the central spot on the axis. During operation, the user can release the handle after pushing the handle towards either side of the translation axis and it will return to its home position.

All state information is gathered by the built-in ESP32 dual-core micro-controller. High-level embedded control and communication is done by this device. For the purpose of this work, it is most relevant to know that it can both stream data via a CAN bus to an Ixxat V2 CAN-to-USB device(*HMS Industrial Networks, Halmstad, Sweden*), as well as distribute linear and angular motor commands to the motor controllers when receiving instruction for those over the same bus. The state is updated at this CAN bus at 100 Hz and that data is streamed via the Ixxat V2 to the PC, for which the specifications are displayed in 3.1.

Motherboard	Asrock X570M Pro
CPU	Ryzen 9 5900X 12-core @ 3.70 GHz
GPU	GeForce RTX 3070 Ti GAMING X TRIO @ 1830 MHz / 8GB RAM
Storage	Kingston Fury Renegade 1TB SSD
RAM	64 GB DDR5 RAM
Operating system	Ubuntu 22.04 LTS

Table 3.1: Hardware specifications and operating system used for simulation.

3.3 Software architecture

The software package delivered for this thesis consists of several Python scripts that configure and dynamically control a SOFA simulated scene. It is built to be modular in the sense that parts such as CAN communication, pathing, visual feedback and (elements of) the control algorithm can be turned off easily. The main script for the framework is `simulate.py`, which imports the simulation controller in `controller2instruments.py` and the scene objects from `templates.py` and `utils.py`. The simulation controller imports the other pathing and control-related classes from `pathing.py` and `CANBus.py`. This hierarchy is also shown in fig. 3.3.

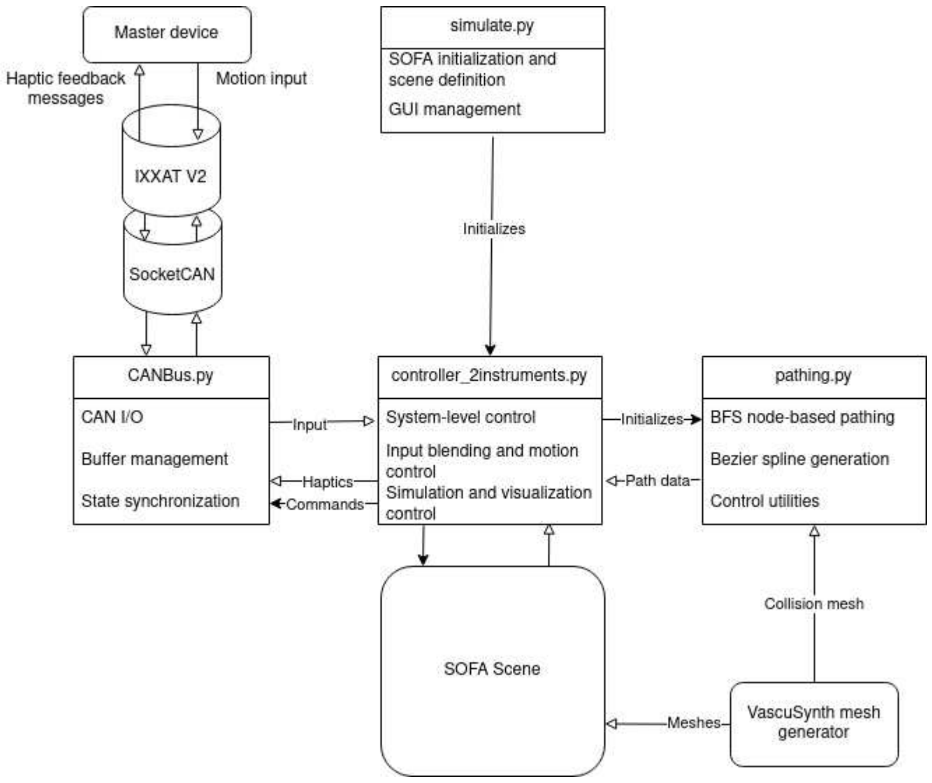

Figure 3.3: High-level software hierarchy of the framework. Black-tipped arrows indicate the flow of commands and initialization, white-tipped arrows indicate the primary flows of data. Not shown are `templates.py`, `utils.py` and `config.py` as they fulfill mostly passive functions or are called only for pre-processing. An alternative module, `controller.py` is available for the simulation of only a guidewire instead of both instruments.

The code is written with an object-oriented approach in mind, and runs primarily on the classes defined in these scripts. The **Controller** class in the `controller2instruments.py` script inherits from the `sofa.core.controller` class built in the SOFA Python API. This class manages all communication to and from the simulator, and contains both high and low level controllers for framework functionality and motion control. The

Path class in the `pathing.py` module takes in a mesh when called by **Controller** and computes a path from user-given input. The **CANBus** class in the script of the same name handles all communication with the master and is written on top of the Bus class in python-can [38].

3.3.1 Python-CAN pipeline

For the reception and transmission of data via the Ixxat V2 to and from the master, a high-performance python module for CAN communication is a requirement. For this purpose, the python-can library was applied to create a virtual *SocketCAN*-driven bus. SocketCAN allows us to use the built-in Linux drivers after setting this up with only a few commands. This allows for some debugging functionality directly from the console as well as a basic but reliable API in python for reading, writing and filtering messages.

In python, the messaging structure is set-up via the **CANBus** class. It provides a digital copy of all states of the master to be polled at any point by the simulation controller. The class makes use of several important attributes of the python-can library. Specifically important is the **BufferedReader**, a subclass that implements a message buffer on the software side. This is combined with a built-in message buffer on the hardware side, and another buffer built-in the kernel driver to provide a steady buffer that can even be read out perfectly when the polling of the queues is delayed significantly.

Additionally, the **CANBus** class implements a filtering option on message arbitration ID's. This is important because in a CAN network, all connected devices receive all messages. This means that all motor commands are equally received by the PC framework, and efficiency is improved by filtering before the queue is polled to prevent unnecessary I/O operations. Furthermore, it is important to account for the very inconsistent sampling rate of the pipeline. CAN does not inherently synchronize messages, and no reliable timestamp for the motions is therefore directly available. What is used instead, is a specially designed buffering system that combines the position with the timestamp at the kernel at the moment of reception by the bus.

To best suit the purpose of controlling the simulated surgical instrument and minimize the amount of messages that actually need to be processed by the bus, it is important to compose all relevant data in a few efficient messages. This is done both by the embedded controller in the master device and the `CANBus` module according to the format detailed in 3.2. The state in this instance is redefined from its definition at the start of this section, to use a single number for the four possible combinations of the grip- and instrument states.

ID	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	Sender
0x32	Linear data		Rotary data				State	Unused	master
	signed short		signed long				-		
0x30	Unused							Acknowledgement	Both
	-							raw	
0x33	Haptic scale		Unused						PC
	signed short		-						

Table 3.2: CAN message format examples of the most important messages for mutual acknowledgement of devices, data transmission and haptic feedback. All messages are little-endian.

The CAN pipeline was qualitatively evaluated during developmental use and user trials, and it was found to run reliably and without noticeable latencies except those caused by the frame rates in the simulator. Back-end queueing is reliable handle by the Socketcan module, and no messages are missed even during significant delays. The class is furthermore designed to be modular, and further messages can be added at any point as well as other sending or receiving devices.

Chapter 4

SOFA Modelling

This section describes the physical modelling effort done in SOFA to simulate the navigation of endovascular instruments on a patient anatomy.

4.1 Instrument modelling

In the endovascular operating room, two primary tools are available as part of a catheterization set-up: the catheter and the guidewire. A schematic example of these instruments is shown in fig. 4.1. This document focuses on the modelling, trajectory planning and control of the distal ends of both these tools. The instruments are modelled from approximately the waving interruption of the tool diagram onward towards the tip. Hub and strain relief, as well as the actuation of the tools outside the patients body are not considered. To make the simulated instruments as faithful to reality as possible, parameters were taken from the devices in the earlier work of Dr. Dagnino and Dr. Kundrat [39]. For the catheter the *Beacon Tip 5 Fr VanSchie2* by *Cook Medical* was modelled, and for the guidewire the *Radifocus Guide Wire M.035"180cm Angled* by *Terumo*. Exact dimensions and material parameters used for the simulation are listed in the appendix table A.1 in Appendix A. As some material properties are not publicly available, material properties from the listed materials were taken from within the ranges in the MatWeb database [40], with the mean stiffness of nitinol for the guidewire, and a stiffness on the low end of the scale for the braided nylon catheter.

The instruments are modelled using the BeamAdapter plugin based on the Kirchoff-Cosserat model discussed in the literature section. The guidewire itself is composed of two parts: a tip part that is more flexible, and a more solid part that provides solidity. Although some guidewires use different materials in the curved end of the tip

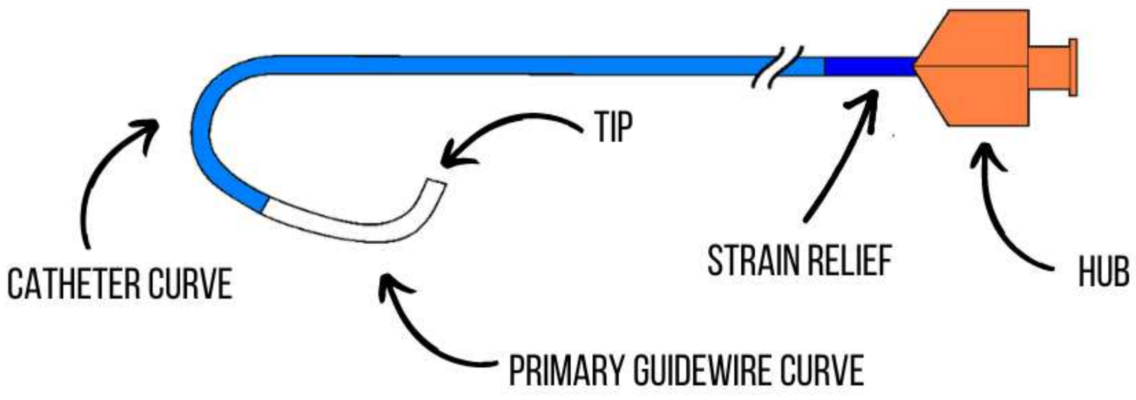

Figure 4.1: Schematic diagram of a gold standard catheter and guidewire configuration.

for extra flexibility, the Radifocus wire that is modelled only has a tapering towards the end, which is represented by a slightly lowered stiffness in this part. The full guidewire is modelled by the BeamAdapter plugin, which also allows for this distinction as long as a discrete amount of beams is placed in either part. The catheter model does not make a specific distinction between material properties in the tip and the rest of the body, as is the case for the Terumo instrument that is modelled.

A multi-model approach (fig. 4.2 is used for the implementation of the instruments in SOFA with six distinct but intertwined models. These are the two *topology* models and the two *visual* models, the mechanical model that combines both instruments and finally the collision model of the combined instruments.

The instruments are modelled as elastic structures, and as such have a rest shape that is returned to when no external force is applied to them. The topology models define this rest shape of the instrument as well as its elastic and material properties. The model is primarily a static container of properties as well as a MechanicalObject object that contains the relative velocities of the instrument edges, without a direct visual representation in SOFA. The rest shapes of both devices are defined by curvature radius, arc length along the curvature and height increase per rotation in case of three-dimensional shapes. The properties implemented in this node are fully configurable in the `config.py` file and visible in App. A table A.1.

The mechanical model, found in the node *InstrumentsCombined*, is where all the state changes are calculated and stored. Here, the points between the edges of the instruments are computed using Bézier splines and all state data is stored as well as the solvers (implicit and linear) that provide this. It also houses the computa-

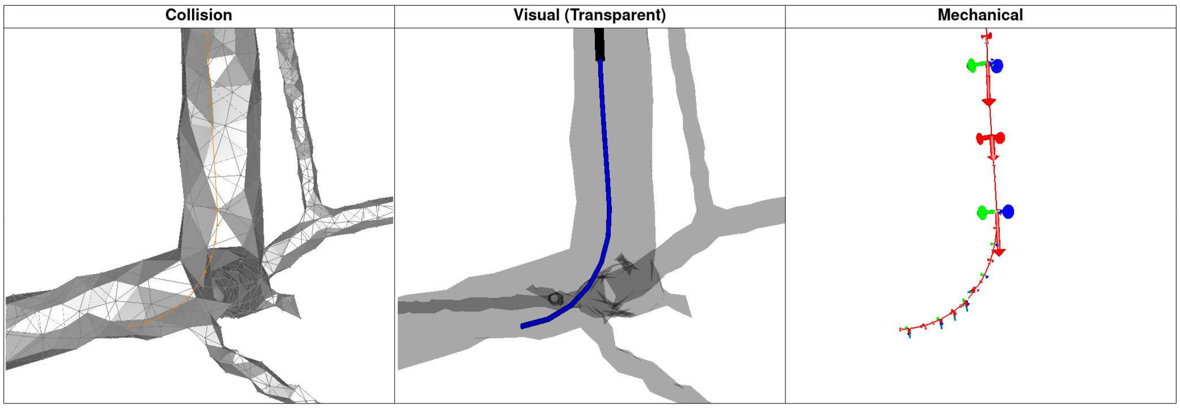

Figure 4.2: Three models of the catheter and guidewire inside the vascular anatomy. Left: a coarser triangular collision mesh for the anatomy. Middle: a finer visual mesh of both instrument and anatomy. Right: a mechanical thin-wire model of the instruments showing the reference frames for each beam

tion for the internal forces, all static constraints and the constraint correction method.

Like the topology models, the visual models of both instruments are very similar to each other. The visual models contain a mapping from the 1-dimensional edges of the topology to the quadrilateral elements used for the visualisation. Colors and visual materials are chosen to match real catheters and guidewires where possible. Although current visual models are rudimentary, it is possible to exchange these at any point without impact the dynamics of the simulation.

Of great importance for the behaviour of the instruments inside the vascular structure are the collision models and constraint handling in the simulator. SOFA allows for two general types of constraints; projective constraints for relatively simple applications, and Lagrange constraints for more complicated constraints as detailed in subsection 2.1.2. Generally, contacts in this work are modeled with Lagrange constraints. At every timestep, SOFA checks if pairs of object will collide using its two-step collision algorithm. The first step, called the broad phase, just checks if the two objects are actually near each other by comparing rectangular bounding boxes around both objects to see if they collide. If the bounding boxes indeed collide, the next step is the narrow phase. This narrow phase checks if the actual collision meshes, in the case of the instruments a set of lines, come within a preset distance (the alarm distance) from eachother. If these meshes indeed collide, the intersection method is called. The intersection method checks if the objects in the narrow phase are actually colliding, and determines what the contact points are. In this model, the *LocalMinDistance* is applied. This method refines the contact points by filtering the contacts through a cone consisting of the orthogonal planes of neighbouring surfaces and invalidates all contacts outside these cones. When the points of contact are determined, a contact response is called that depends on the direction of the collision. In the case of this model, this is a friction response that applies a force aligned with the direction of impact. An example of how the output of this pipeline is rendered in SOFA is shown in fig. 4.3.

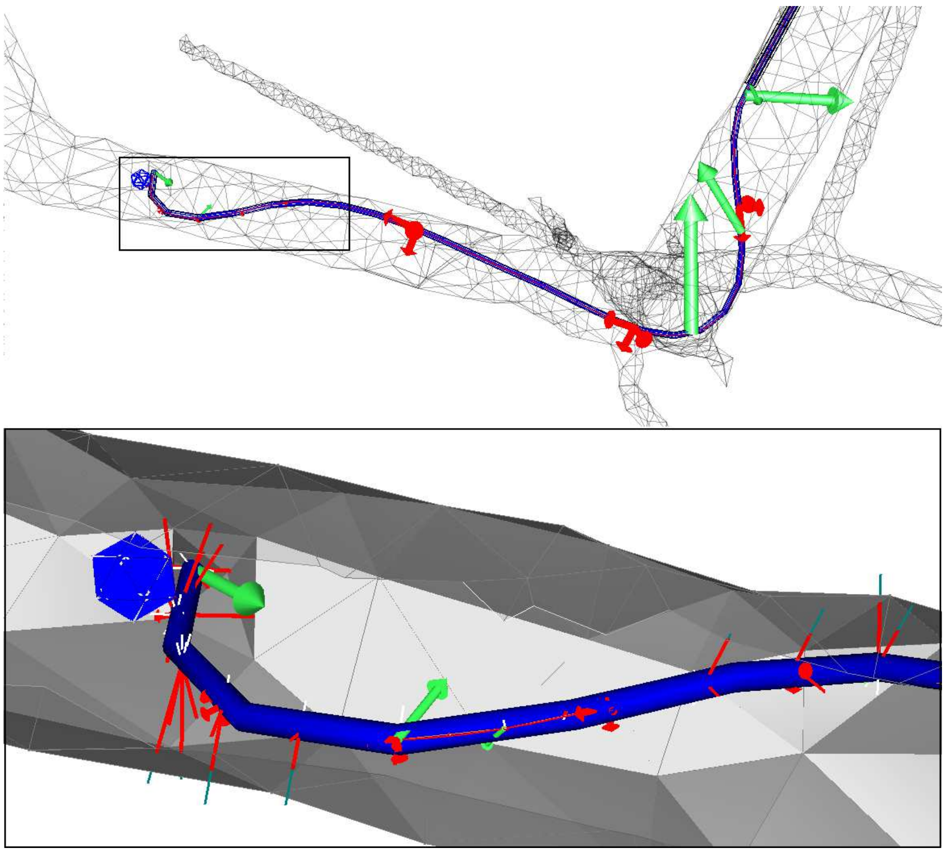

Figure 4.3: Top: wireframe depiction of the two instruments in a very tortuous state undergoing a collision. The image shows a visual mesh together with the computed interaction forces and visualization thereof. The bottom image contains a detail of the colliding section of the tip. Several visual elements are present in the picture. The guidewire instrument is visible in blue in both images, with the catheter in black. Red lines indicate that the *alarmdistance*-perimeter is entered by the instrument, and that it is being monitored for a response. Inside the instrument, red axes are slightly visible that form the reference frames for the interaction forces. Lastly, the blue sphere indicates the closest nearby point on the path.

4.2 Vascular model

Besides the operating instruments, the modelling of a suitable environment is also necessary. While there are many specific arteries where vascular surgery is typically performed, the goal of the simulator is to provide a broad framework and facilitate control testing as well as training. It is therefore of importance not to just model anatomy, but to be able to test in a broad spectrum of environments with parameters that can be optimized for the specific requirements. This can be an anatomical model obtained from magnetic resonance imaging for instance, but it can also be generated within this framework by the integration that is supplied to the Vascular Synthesis toolkit [37]. This package is used to randomly generate vascular structures according to the number of bifurcations and a map of oxygen needs in an area, as shown in fig. 4.4. This allows for a wide range of possible 3D structures, which is very useful specifically for navigation and control training and development. Of course, real surgical training will require more anatomical maps, which are both partially supplied by SOFA, available from patient-specific data, and widely available on the web.

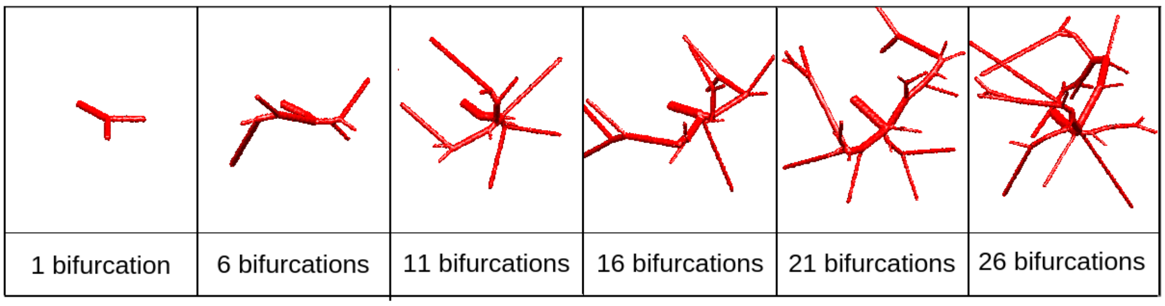

Figure 4.4: VascuSynth output volumes with several numbers of bifurcations on random oxygen demand maps.

The VascuSynth generator is based on the Insight Segmentation and Registration ToolKit (ITK [41]), a well-developed open-source framework for working with medical imaging and its products. With the use of ITK's Python extension, it is therefore possible to import the vascular meshes to the SOFA environment using the pipeline depicted in 4.5.

For the interaction with the virtual anatomy, SOFA offers the possibility to use flexible structures. However, in this case the choice is made to use a rigid material for several reasons: the effect of the elastic forces in the walls are deemed to be of low consequence for the navigational properties of the instruments, and there is

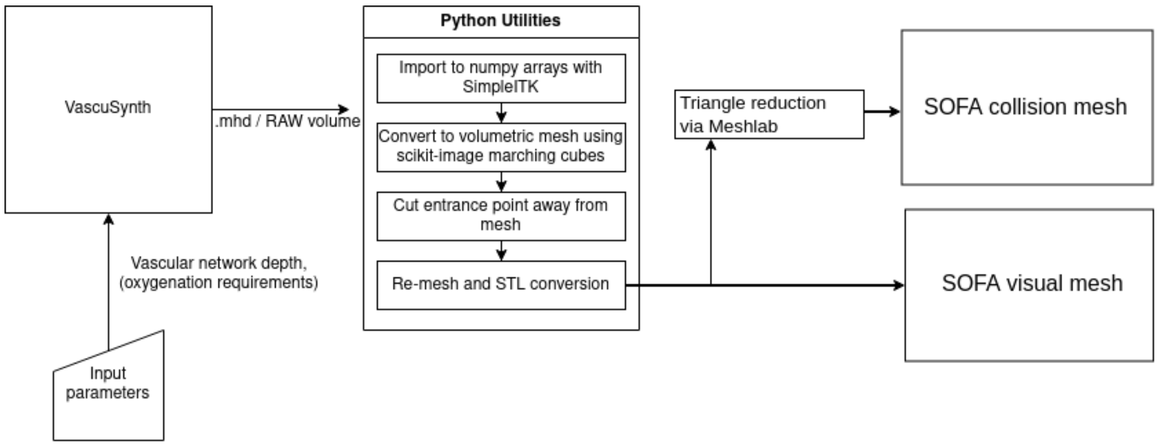
VascuSynth
.mhd / RAW volume
Python Utilities
Import to numpy arrays with SimpleITK
Convert to volumetric mesh using scikit-image marching cubes
Cut entrance point away from mesh
Re-mesh and STL conversion
Triangle reduction via Meshlab
SOFA collision mesh
SOFA visual mesh
Vascular network depth, (oxygenation requirements)
Input parameters

Figure 4.5: Schematic of supplied pipeline for utilizing the VascuSynth databese and generator.

insufficient data available to validate the realism of the effects in SOFA. Secondly, the great computational expense that comes with realistic flexible surfaces might be prohibitive for any hopes of real-time application of the framework. For these reasons, a simple triangular rigid mesh is used to model the vessels. The primary visual mesh used for testing uses 4970 triangles for a tree with 12 accessible final branches, an 80% reduction from a direct one-to-one mesh import. For the collision mesh, a secondary mesh is used that is a direct reduction of the visual mesh with another ~80%.

Chapter 5

Pathing and shared control framework

5.1 Pathing

To start the pathing sequence, the anatomical model is skeletonized using a wavefront algorithm. This algorithm is adapted from the Skeletor plugin [42] and works by propagating a wave through the structure starting in a single point. If the wave hits vertices that are both connected and have the same distance to the seed (i.e., source) of the wave, they form a ring which can be collapsed to a single point. These point are then gathered to form the vertices of the skeleton. The benefit of this algorithm is that it is specifically well-suited for tubular structures like blood vessels because these are naturally ring-shaped with a centerline through the middle. By tuning this algorithm, we can also extract the radii of the wavefronts at every ring, giving a valuable indication of the width of the tubes. This information is further refined by applying a ray-casting algorithm [43] at every vertex to obtain the minimal, maximal and median distances to the wall.

The skeletonization algorithm gives us a tree-like structure of nodes and distances (costs) between nodes. While this does not take into account the space within the vessels, it allows us to perform the first step: finding the global path. For this purpose, a breadth-first search algorithm is applied to the tree. This algorithm is shown in the pseudo-code block (Listing 5.1) below.

```
def BFS(startnode, endnode)
    while not queue.empty():
        if current_node = endnode:
            path_found = True
            break

        for next_node, cost in neighbors[current_node]:
            if next_node not in visited:
                queue.put(next_node)
                parent[next_node] = current_node
                visited.add(next_node)

    if path_found:
        path.append(endnode)
        while parent[endnode] is not None:
            path.append(parent[endnode])
            endnode = parent[endnode]
        path.reverse()
```

Listing 5.1: Pseudo-Python code of the BFS algorithm

The algorithm listed delivers the only path in a tree-like structure such as the one we are considering. If there would be multiple possible paths, it could easily account for path cost by comparing the total costs in the final step and selecting the cheapest path. Because it is an exhaustive algorithm, it is best applied to discrete graphs that have no circular path or self-intersections. The algorithm works in this case because we are considering a discrete and compact tree, but is chosen for completeness and reliability. It is not the fastest possible method but since it only needs computation once before the simulation starts, its computation time of 4 ms for a 400-vertex skeleton is negligible when compared to the 250 ms skeletonization duration for the same mesh.

When the first stage of the path is completed, a discrete path from beginning to end is available. This discrete path consisting op nodes d_i ($i = -2, -1, 0, 1, ..L$) is then used as the start for the generation of the control points for a continuous Bézier-spline trajectory that is C^2-continuous based on [36] following eq. 5.1 for the initial and final control points p_m. In this equation, L is the number of cubic Bézier curves, which is the total amount of nodes on the path reduced by 2.

$$\begin{aligned} p_0 &= d\_2 & p_1 &= d\_1 \\ p_2 &= \frac{1}{2}d_{-1} + \frac{1}{2}d_0 & p_{3L} &= d_L \\ p_{3L-1} &= d_{L-1} & p_{3L-2} &= \frac{1}{2}d_{L-2} + \frac{1}{2}d_{L-1} \end{aligned} \tag{5.1}$$

Eq. 5.2 is used for all subsequent control points, with $i = 1, 2, ..L - 2$:

$$\begin{aligned} p_{3i+1} &= \frac{2}{3}d_{i-1} + \frac{1}{3}d_i \\ p_{3i+2} &= \frac{1}{3}d_{i-1} + \frac{1}{3}d_{i-1} + \frac{2}{3}d_i \\ p_{3i} &= \frac{1}{2}p_{3i-1} + \frac{1}{2}p_{3i+1} \end{aligned} \tag{5.2}$$

This results in a series of control points that can be used for the generation of a continuous path. This path is then interpolated using the Bernstein polynomials as detailed in eq. 2.7. As the path generated by a Bézier curve has the property of never exceeding the convex hull of its control points, the strategic allocation of the points can prevent the curve from passing outside the modeled mesh. Because the discrete path only contains higher concentrations of points on the curves and fewer points on the straight sections, there is a built-in tendency to stay within the confines of the mesh. This is however not guaranteed in this stage, and in rare circumstances the path can fall outside the mesh.

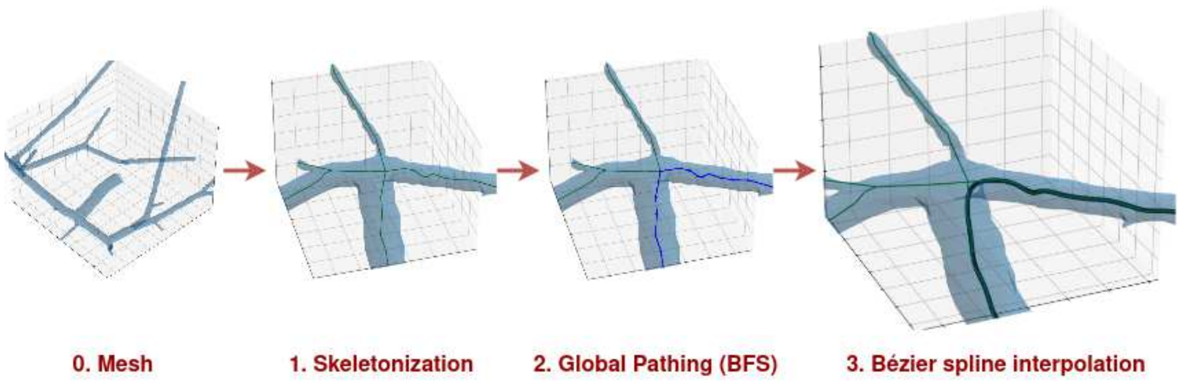

Figure 5.1: Primary stages of the pathing algorithm. Stage 0 shows the STL mesh extracted as described in subsec. 4.2. Stage 1 to 3 shows the application of the pathing algorithm described in this chapter. All path images are 2D projections of 3D splines in matplotlib, and might convey slight inaccuracies.

5.2 Shared control of endovascular instruments

With the pathing information from the previous section, it is now possible to implement a shared control algorithm to optimally employ operator skill while at the same time implementing important failsafes. This is done by implementing dynamic guidance for the instruments based on the guidance factor g. When g is 1, the system operates at the highest level of guidance, and will be quick to provide inputs. On the other hand, when g nears zero, the operator is left completely to his/her own devices. The current implementation of shared control is partially discrete: enabling certain features at set intervals and partially dynamic, scaling continuously with the set level of g.

5.2.1 Attitude control

To find the optimal instrument attitude, a set of equations is composed to link the guidewire attitude and shape to its alignment with the current path. These equations assume that the angle of axial rotation, θ, is used as an input and that the primary tip parameters (inner diameter d_{tip} and arc length L_T), and the positions of the instrument body are known.

Three reference frames are defined for this purpose: the world frame ($\boldsymbol{F}_W$), (instrument) body frame ($\boldsymbol{F}_B$) and tip frame ($\boldsymbol{F}_T$) (see fig. 5.2). The world frame is the simulator coordinate frame, the body frame is always aligned with the straight part of the guidewire body at the point right before the curve. The tip frame is always aligned with the direction the last tip edge points towards. The control objective is alignment between θ and the negative y-axis of $\boldsymbol{F}_T$. This alignment is quantified using the vector difference between the tip frame and the closest path edge, both normalized to unit length. We assume that α and L_T are considered as static parameters in this case. In reality, α is not completely static; specifically, it is influenced by the stiffness of the catheter instead of the guidewire when the catheter slides over the former instrument. Additionally, the angle is influenced by the collisions the guidewire makes with the environment if the collisions happen very close to the tip. The first case reduce the magnitude of α, but, for a frontal collision, does on its own not change direction of the tip in the body x- and z-axes significantly. The latter case is a weak point in the controller, and cannot be solved with the current inputs and state information, as predicting the dynamics of these collisions is not computationally feasible in a controller.

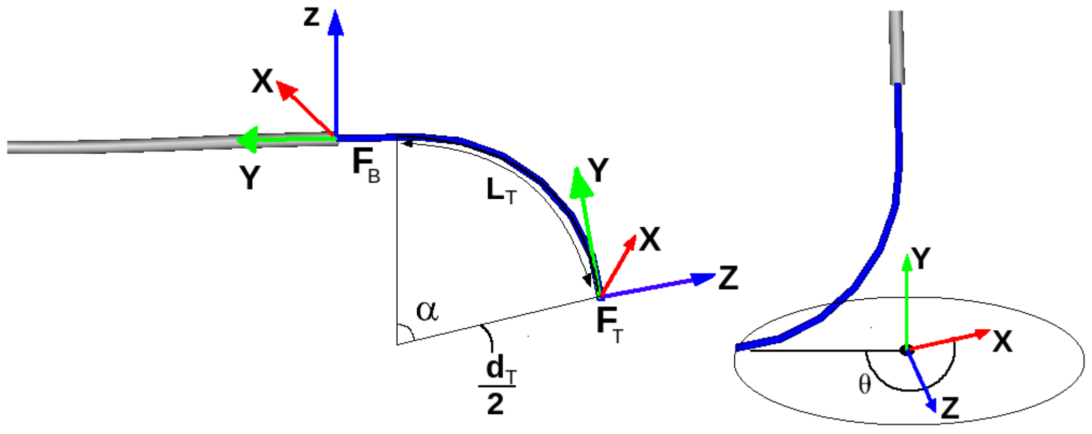

Figure 5.2: Reference frames and kinematics of the guidewire instrument

Within the body frame of the instruments, we can quantify the outer point of the tip, $\boldsymbol{p}_{tip}$ with the equation below:

$$\boldsymbol{p}_{tip} = \begin{bmatrix} x_{tip} \\ y_{tip} \\ z_{tip} \end{bmatrix} = \begin{bmatrix} r_{curve}(1+\cos\alpha)\cos(-\theta+\frac{\pi}{2}) \\ -\frac{d_{tip}}{2}\sin\alpha \\ -r_{curve}(1+\cos\alpha)\sin(-\theta+\frac{\pi}{2}) \end{bmatrix} \tag{5.3}$$

Where r_{curve} is a parameter that expresses the radius of the circle the tip of the instrument tracks when rotating around the body y-axis:

$$r_{curve} = \frac{d_{tip}}{2}(1+\cos(\alpha)) \tag{5.4}$$

To compute the error with the path, the path direction $\boldsymbol{v}_p$ should also be converted from the world frame to the body frame. This is done using rotation matrix ${}^{W}\boldsymbol{R}_B$ which is extracted from :

$${}^{B}\hat{\boldsymbol{v}}_p = {}^{W}\boldsymbol{R}_B{}^{W}\hat{\boldsymbol{v}}_p \tag{5.5}$$

Finally then, the control problem is formulated as:

$$e = {}^{W}\boldsymbol{R}_B{}^{W}\hat{\boldsymbol{v}}_p - \frac{p_{tip}}{|p_{tip}|} \tag{5.6}$$

Which allows for minimization using least-squares optimization via the Broyden-Fletcher–Goldfarb–Shanno (BFGS) algorithm on the error function in eq. 5.6. The angle computed here is then reached by the instrument through a proportional controller. In a real environment, additional control steps may be necessary due to the actuator and/or wire dynamics. As realistic actuation is considered outside the scope of the project, this is however not considered in this document. The step response of the controller is shown in fig. 5.3.

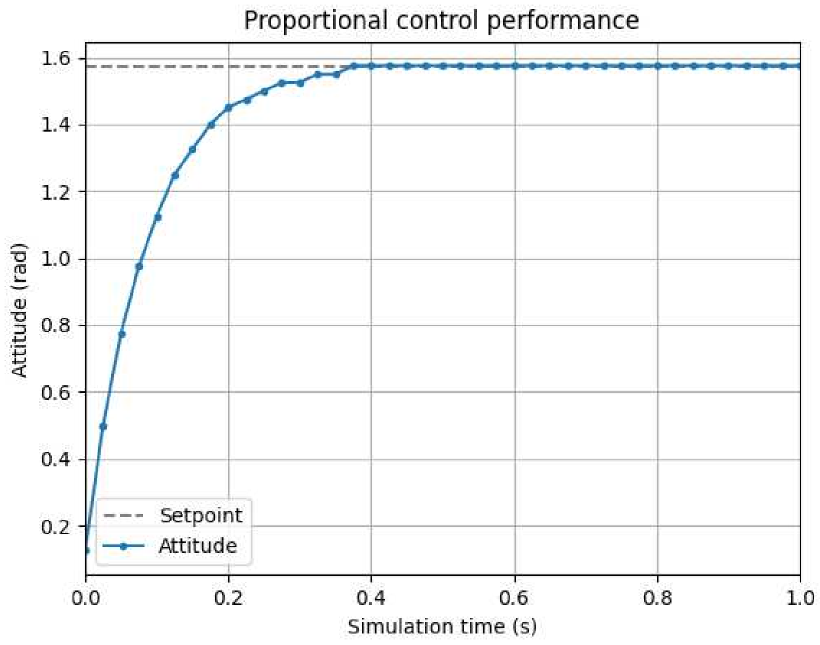

Figure 5.3: Step response of the unobstructed attitude controller in SOFA. Rise time: 0.375s, Steady state error: 0.025.

5.2.2 Force trauma prevention

Additionally, steps are taken to mitigate one of the most dangerous causes of injury in endovascular operation: puncturing the vascular walls [44] . For this purpose, a force-tracking algorithm is implemented in SOFA. This function listens in on the constraint solver inside SOFA to extract the forces. At every time step, a constraint matrix is generated by the Lagrange solver. Each row of the matrix $\boldsymbol{H_j}$ is shaped like displayed in table 5.1, and contains an N_c amount of constraints for constraint number n_c, constraining points $i_1, i_2, ..$ in the direction given by $\begin{bmatrix} x_1 & y_1 & z_1 \end{bmatrix}^T$.

n_c	N_c	i_1	x_1	y_1	z_1	i_2	x_2	y_2	z_2	i_3	x_3	y_3	z_3
1	3	50	0.29	0.30	-0.15	51	0.41	0.19	-0.03	52	-0.21	0.24	0.76

Table 5.1: Example of constraint matrix from Lagrange constraint solver

The magnitude of the forces is given by the Lagrange multiplier, following equation 5.7. In this equation, $\boldsymbol{A}\delta\boldsymbol{v}$ represents the mass matrix and accelerations, $\boldsymbol{F}$ the constraint forces, h the size of the time step, b is part of the system matrix which is near-constant for small time steps as detailed in [21], and $\boldsymbol{\lambda}_f$, the unknown Lagrange multipliers.

$$\boldsymbol{A}\delta\boldsymbol{v}\boldsymbol{F} = \boldsymbol{b} + h\boldsymbol{H}^T\boldsymbol{\lambda}_f \tag{5.7}$$

For resolution, the Lagrange multipliers are extracted from the Gauss-Seidel algorithm (See sec. 2 , [21] [22]), and multiplied with $\boldsymbol{H}$ to get the constraint forces per indexed point.

As the purpose of the controller is to provide assistance to the operation in safely conducting the operation, haptic feedback is implemented using this information. The constraint forces at the tip are summed to obtain a haptic force, which the user feels through the operation of the handle of the master device. The master device uses a discrete admittance controller based on a virtual mass-damper system [45]:

$$X_d(k) = \frac{X_d(k-1)[2m - bT] - mX_d(k-2) + T^2[-f + f_d]}{[m + bT]} \tag{5.8}$$

where m and b are parameters for inertia and damping, respectively, k is the discrete time step, and X_d is the position. Because stability and performance during operation is heavily linked to the carefully tuned damping parameter, only the inertia parameter is somewhat freely settable. This inertia parameter can be scaled from 100 to 2000% of its base value, where 100% represents only the instrument inertia and constant resistance encountered when navigating the blood vessel. This parameter, the mass ratio r_m, is set through a CAN message once per simulation loop. It follows the adjustment law in eq. 5.9.

$$r_m = C_h \sum_{-10}^{0} \frac{F[k-1] + F[k]}{2} \Delta t \tag{5.9}$$

In other words, the virtual mass scales with the integrated contact force $\boldsymbol{F}$ and a scalar haptic constant C_h. The haptic feedback is then implemented such that it only acts when the user is trying to advance the device, but not when retracting. Furthermore, the force is only experienced when advancing the instrument which is currently most extended, and thus forms the tip of the combined instruments. This is measured by comparing the curvilinear distance between the most advanced points of either instruments, $x_{tip,C}$ and $x_{tip,G}$ for catheter and guidewire respectively. The mass ratio is then determined by:

$$m = \begin{cases} m_0 r_m, & \text{if } 1 \leq r_m \leq 20 \text{ and } \delta x_{tip} > 0 \\ 20 m_0, & \text{if } r_m \geq 20 \\ m_0, & \text{otherwise} \end{cases} \tag{5.10}$$

where $\delta x_{tip} = \sigma_{ci}(x_{tip,C} - x_{tip,G})$ with σ_{ci} the sign of the currently controlled instrument; 1 and -1 for the catheter and guidewire respectively. The mass ratio, or the scaling of the haptic feedback is based on [46], with the threshold for haptic feedback at 0.011N, the amount for a moderate contact force, and the maximum amount of 2000% at 0.22N.

Force visualization

Not only the contact forces on the tip of the instrument are of importance for the safety of the patient. Coiling of the instruments, friction against the vessel walls and built-up tension are all examples of dangerous situations that can be prevented by adequate force sensing and management. It is therefore useful to provide visual feedback to the user when he or she is applying a lot of force, leading to possibly unwanted situations. On the other hand, as the navigation of the instruments depends on friction forces, and taking some corners might not be possible without the application of a measured amount of force, it is undesirable to force the hand of the operator too much in this regard. Because of this, a combined of visual feedback for strain and friction forces combined with the more forcing approach described above for the interaction forces in the tip was chosen as most appropriate. The forces are rendered as arrows in the location of their occurrence as shown in fig. 5.4. The forces scale both in color and size: higher forces are displayed larger and redder. The scaling for the forces is based on [46], with no hard limit on size but maximum redness at 0.04N, twice the found mean strong force of 0.02N.

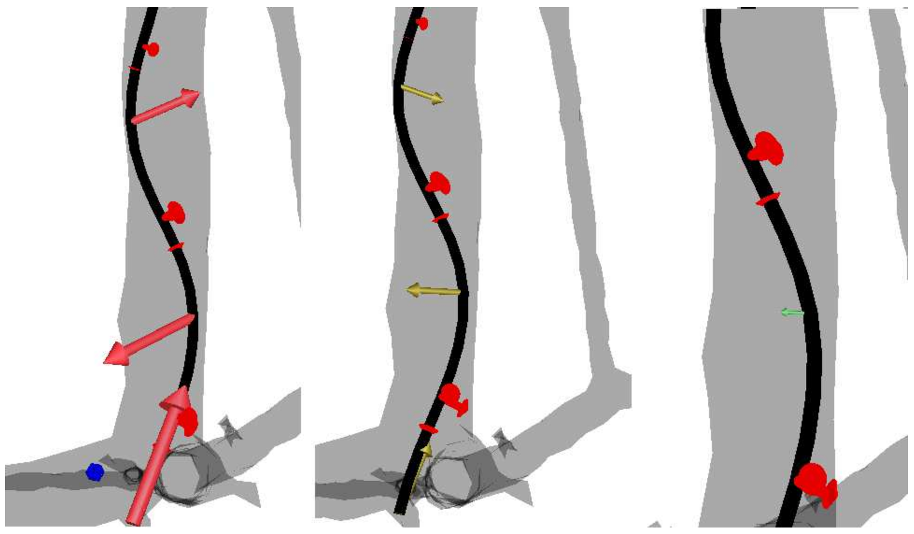

Figure 5.4: Force visualization on the SOFA visual model. Left: high stress forces are shown in red. Middle: medium stress forces show yellow. Right: low stress forces are green.

The figure shows that the forces (colored arrows) are rendered in according to the expectations. Normal forces are shown where the internal forces press the instruments against the vessel walls during a curve, and the tip forces are shown in the most distal part where it slightly presses the vessel wall. Forces due to actuation are not shown, and at every edge, only the resulting force is shown. Gliding friction forces are rendered, but often small in comparison to normal and tip forces.

5.2.3 Dynamic guidance

When the user starts an operation, he/she is asked to set a level of dynamic guidance. This guidance parameter, g, decides the amount of assistance the user will receive during the operation. If necessary, g can be changed during the operation with keyboard shortcuts. The guidance level scales from 0 to 1. At all levels, visual force feedback is enabled. Shared attitude control is enabled in a scaling capacity at any level above 0, and haptic feedback starts scaling from level 0.25.

The strength of haptic feedback scales via the mass ratio r_m (see eq. 5.10 as well) with the guidance parameter g:

$$r_m = C_h * g * \boldsymbol{f}_{tip} \tag{5.11}$$

with C_h being a scaling parameter determined with a heuristic approach and $\boldsymbol{f}_{tip}$ as the tip force.

For the angular velocity output y, the shared controller combines the user rotational input with the controller output via a weighted blending function:

$$y = \frac{w_u x_u + w_c x_c}{w_u + w_c} \tag{5.12}$$

This function combines user input, x_u, and controller input, x_c with a set of two weights based on the guidance level, w_u and w_c to deliver the final output angular velocity for the guidewire. It is generally not necessary to apply attitude control to the catheter as it is will be axially aligned with the guidewire and is not used for navigating corners.

The weights of the input blender are determined by the guidance parameter as:

$$w_c = g \quad w_u = 2 - g \tag{5.13}$$

This scales the amount of influence of the user back from 100% at 0 guidance, to 50% at 1.0 guidance. This means that the user can override the controller easily at low levels, but can only stop the controller at maximum effort at the highest guidance level.

A similar algorithm is implemented on the translation axis when the user starts to experience haptic feedback. While in earlier stages a hard stop when exceeding a threshold was implemented, this can sometimes be prohibitive to the use of the instruments for legitimate purposes and can frustrate the operator. Therefore, in situations where strong tip forces are experienced, the input translational velocity are mitigated. Note that these velocities are measured at the base, and as such translate to a pushing force through built-up tension at the distal end:

$$v_i = \begin{cases} \frac{w_{uv} v_u + w_{cv} v_c}{w_{uv} + w_{cv}}, & \text{if } v_u \geq 0 \\ v_u & \text{otherwise} \end{cases}$$

In this equation, w_{uv} and w_{cv} are the translational weights for the user and control inputs respectively. In this case, the control input is set to zero when the haptic feedback is enabled, and the weighting scales with the level of haptic feedback applied and the guidance parameter: $w_{uv} = 1$, $w_{cv} = \frac{r_m}{100} * g$. In other words, at the maximum level of guidance and haptic feedback, the user will only achieve 5% of the normally achieved input velocity. Retraction, or moving the non-colliding instrument is always uninhibited. In these cases, the user input $v_i = v_u$.

Chapter 6

Evaluation

6.1 Methodology

For the quantitative evaluation of the implemented algorithm as well as a qualitative evaluation of the broader framework, an experiment is set up containing a total of 32 trials spread over 3 operators performing 2-4 trial sets of 4 trials each. One participant, P1, completed 4 separate trial sets for all guidance levels, all other participants completed 2 trial sets per guidance level. In addition to this, a short training period before the first trial is permitted. The trajectory that is used for the participant is displayed in fig. 6.1. For each trial, the same anatomy mesh and trajectory are used to keep the conditions similar for all users. Guidance levels were set to 0, 0.25, 0.66 and 1.0. The first series of trials was conducted in that respective order. The second and, if present, further trials were conducted in randomized order.

An earlier run of experiments was also conducted, containing 5 participants over 24 trials. From these experiments, improvements were made to the logging, performance of the framework and control operation. The data presented in this chapter is only from the recent set of 32 trials, although some lessons from the first series are briefly mentioned.

Some additional visual elements for the guidance of the operator are included in the simulation environment. These are two markers for the nearest point on the path (blue) and the final destination for the tip of the instrument (red). The participants are instructed to maneuver the instruments such that at least the guidewire touches the endpoint. For evaluation purposes, any trial where the tip of both instruments fully passes **C2**, the trial is considered complete. For any timing measurements, this is counted as the endpoint.

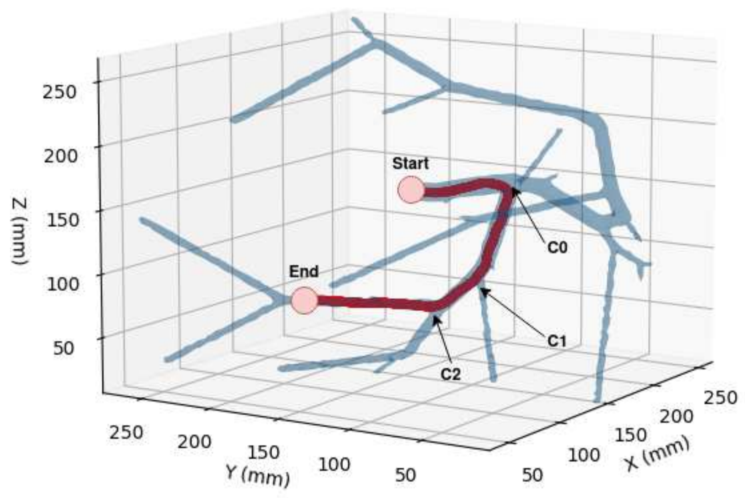

Figure 6.1: Trajectory for the evaluation trials showing the start (entry) and end-points in the mesh. **C0**, **C1** and **C2** mark the major corners during the trajectory.

The following data is logged during the experiment:

- **Trial information**: Name, trial number, guidance level, endnode
- **Time**: Absolute time, simulation time
- **Position**: All edge positions for the catheter/guidewire combined instrument, relative positions of instruments to each other.
- **Forces**: interaction forces ($\boldsymbol{f}_{x,i}$, $\boldsymbol{f}_{y,i}$, $\boldsymbol{f}_{z,i}$) at every instrument collision edge e_i, $i = 0, 1..60$
- **Master input**: Rotational and translation buffers for both devices, grip state (gripped/not gripped), selected instrument (catheter/guidewire)
- **Master output**: The amount of haptic feedback that is applied
- **Distance to path**: Tip distance to path, closest point on path (see sec. 5)
- **Controller**: Attitude controller setpoint
- **Virtual instrument state**: axial rotation (θ, see sec. 5), catheter velocity v_c, guidewire velocity v_g

Before the start of a trial day, the master device is calibrated via an automatic calibration procedure. This procedure requires the operator to move the handle back and forth, after which the device will perform one full translational cycle as well. This calibration sequence is necessary to adjust the force sensors to the friction in the shaft. The rotational axis sensors does not require any repeated calibration procedure.

There were some anomalies present in the simulation that affected the data and thus required pre-processing. Due to meshing anomalies, very high forces were registered in very specific sections of the trial, primarily when pushing either instrument beyond the first corner. When the catheter instrument is pushed through this corner, the guidewire might be so close against the mesh that there is no space for the catheter to move. This interaction can display forces 1̃00 times the normally measured amounts. For this reason, some sections had to be excluded from the forces statistics. This exclusion is based on video evidence combined with the extreme outliers in the data. This was done for 17 out of 32 trials, excluding a total amount of 3 minutes, 29 seconds from a total experiment duration of 1 hour, 32 minutes and 58 seconds (excluding passive time). No more than one continuous segment was excluded for each video.

6.2 Evaluation of framework and operation

The tests give an opportunity to qualitatively evaluate the framework in terms of performance. For this, several aspects are of importance:

- Computational performance of the framework. This includes average frame rates and frame rate drops during specific actions, as well as time spent processing for sub-steps of the simulation.
- Integration with master device. This includes communication latencies, presence or absence of dropped CAN-frames, subjective responsiveness between master device and simulation.
- User experience and visual feedback. Here, the subjective experience of the participants is shortly discussed from short interview questions that were asked at the conclusion of the experiments. Additionally, the visual feedback is evaluated.

6.2.1 Computational performance

Computational performance of the framework is a bottleneck for good operation in interactive simulation: if the frame rates drop too low this affects responsiveness to input, perceived realism of the situation, haptic feedback and other aspects of the simulation. In the current state, performance was tested on one specific configuration that was selected to have a medium-high amount of bifurcations ($n = 16$), as well as some variation in corridor width. During the trials, the selected collision mesh had $N = 990$ triangles, which was optimized after a considerable decline in simulation rates was observed during the first set of trials ($N \approx 5000$) which was already a large reduction from the original at $N \approx 3.0e4$. For the instruments, rather fine models are selected to model the navigation and kinematics of the instruments well, using two instruments with 200 mechanical edges and 60 collision edges each. Frame rates are heavily influenced by the amount of collision checks and calculations the simulator has to evaluate (see tab. 6.1). With the current parameters, the approximate frame rate is 60 frames per second before the first corner of the mesh, 35 frames per second after the second corner, and steadily declines until approximately 10 frames per second when nearing the third corner. At this point, half of the beams in the instruments are subject to collision detection, with the other half still stored at the start point.

Beside the simulator itself, time is spent on the communication, controller calculations and surrounding python framework. Mesh loading and skeletonization are done before the first time step and take approximately 310ms, while the initial generation of the path takes 30-100ms depending on the complexity of the simulated anatomy. Table 6.1 shows the relative simulation time spent on the sub-steps of the operation as measured on the hardware in tab 3.1, as well as on the framework that is added. Specifically the logging operations are computationally expensive, and form a bottleneck for frame rates before collisions are as relevant. As soon as some collisions are present, the primary bottleneck is the matrix resolution and constraint correction step. Because any interaction force measurement requires explicit numerical solutions to the compliance matrices, this contribution is amplified due to the addition made in this thesis relative to applications that only require interactive force computation within the more optimized SOFA environment.
During the trials, simulation times and rates were logged. This gives an impression of the average simulation rates (simulated seconds per real second), in tab 6.2. The next section also supplies data per participant and guidance level, in fig. 6.8.

Location	$<$**C0**	**C0 - C1**	$>$**C1**
Frame rate (FPS)	55	30	12
Time spent on (%):	100	100	100
Control and communication	53.3 (12.3[1])	33.7 (6.7[1])	20.2 (5.1[1])
Free motion + Collision detection	6.0	4.3	2.9
Build compliance system matrices	10.0	13.9	14
Matrix resolution and constraint correction	19	43.0	60.0
Others	11.7	5.1	2.9

Table 6.1: Performance and time spent on sections of the collision pipeline as detailed in sec. 2. Locations indicate the location of the guidewire tip relative to the corners as shown in fig. 6.1. [1]: percentage in parentheses shows time spent when all logging operations are ignored. This does not influence the relative size of other percentages.

	Comp.[1] time - real (s)	Comp.[1] time - sim.[2] (s)	Sim.[2] rate (s^{-1})
Mean	174.3	121.7	0.719
Standard deviation	53.0 (30.4%)	33.4 (27.4%)	0.139 (19.3%)

Table 6.2: Simulation times and rates: mean and standard deviations. [1] Completion times. [2]: Simulated

6.2.2 Integration with the master device

Operating the master device requires a calibration procedure as mentioned at the start of this chapter. This procedure must be performed daily when the master is used, because it cannot remain powered during the night. This calibration procedure leads to a somewhat inconsistent amount of force expressed during the homing sequence (when the handle is released). As a temporary solution, the calibration procedure is therefore sometimes repeated until the force sensor is not too aggressive. Additionally, due to a fabrication anomaly the photo-sensor that registers the gripped state of the instrument can be inconsistent, leading the homing sequence to initiate before the operator releases the handle. This can be mostly circumvented by instructing the participants where and how to grip the handle to adequately obscure the photo-sensor, as was done during the experiments. Lastly, a rare anomaly can sometimes cause the handle to rotate without prompting, leading to the associated rotation in the simulator.

When frame rates exceed approximately 10 frames per second, the simulator feels very responsive to the master device in both the translational and rotational dimension. If the frame rate declines further, the associated delays are noticeable in the responsiveness of the instruments, but no signs of communication delays beyond those caused by the discrete simulation time-step are observed. The CAN bus receives and sends data at rates of approximately 100 Hz (10ms/cycle) measured on the embedded master controller. Even when the frame rate declines to 2-3 frames per second, the CAN bus runs in a separate thread, and will not miss messages at this point, and as such inputs are eventually fed through. Because the low-level back-end of the CAN-module is run on a system level in Linux, it does not suffer from the Python global interpreter lock (GIL) and can thus handle the essentials in parallel to python processes and SOFA. Because of this, it is unlikely for simulation performance to cause problems in communication latencies and rates, and this was indeed not observed during the experiments or at any other point in development. Likewise, haptic feedback remains consistent on the master device during frame rate drops, and is consistent throughout. The scaling and thresholds for the force feedback are qualitatively determined, and not objectively calibrated against some metric.

6.2.3 Notes on user experience and visual feedback

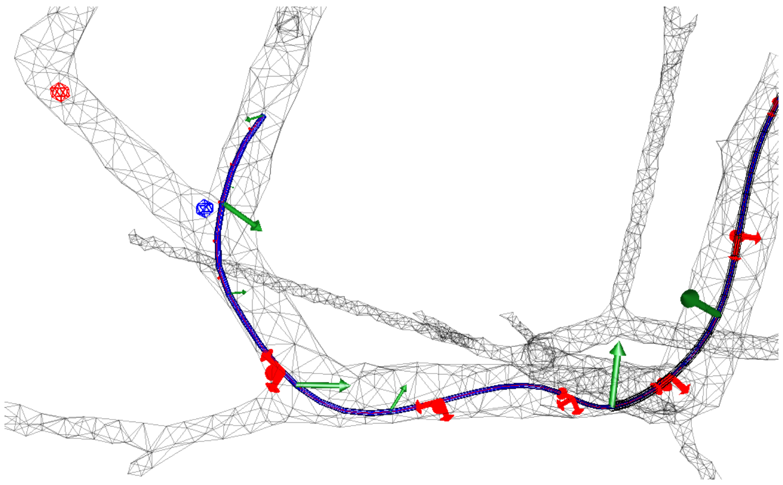

Figure 6.2: Visual aid to the operator during the trials. The blue dot indicates the local setpoint, or the closest point on the current path. The red dot indicates the endpoint of the trajectory. Green arrows indicate forces on either instrument. In this case, the instruments were purposefully put under some tension for demonstration purposes. The representation shown is a *wireframe* representation of the mesh, an alternative visualization also available to the operators, that can give a more clear vision of the borders of the mesh.

Visual feedback as shown in fig. 6.2 was present during all the tests described below at all guidance levels. Unfortunately, color scaling of the displayed forces was disabled during the trials because of performance issues. Some operators gave short feedback in this regard during the interview. One participant noted that the arrows showing the forces in the instruments were helpful in preventing the instrument from coiling up or getting stuck, but that the scaling of the force arrows showed them either quite small (during frontal collisions) or very big (due to built-up tensions).

Another participant noted that the 3D vision in the operation was not always optimal, even though rotation and translation of the camera were allowed for participants. They observed that for some corners, multiple points of visions were necessary. Multiple participants noted some difficulty with the collision mesh, which is coarser than the visual mesh. A participant that also took part in the earlier trials noted that the coarser collision mesh, while allowing for much smoother operation, was harder to navigate because it did not fully coincide with the visuals, and had more sharp corners. This difference in representation sometimes led to a small difference in where the operator observes the limits of the vessel to be, and where exactly a collision can take place. The first participant also noted that the simulator gave a visually convincing representation of the instruments, but that excessive speed or force can lead the instruments to take shapes with unnaturally sharp curves, or lead the instruments to clip outside the mesh, especially during frame rate drops. This clipping can be directly related to the changes that were made after the first series of trials to increase performance; reducing the maximum amount of iterations and error tolerance for the linear solvers.

Regarding the operation of the attitude controller and haptic feedback, users noted that the presence was very noticeable on guidance levels 0.25, 0.66 and 1.0. Both users questioned on the topic noted that they had a personal preference for levels 0.25 and 0.66, 1.0 being too strong an influence. One user that was completely novice to the system noted that the operation is quite difficult without any guidance. The users found haptic feedback clearly present on levels 0.25, 0.66 and 1.0, with no clear preference between the levels. One user noted that once the haptic feedback is enabled, it arrives at a prohibitively high level fairly quickly, and so it was difficult to sense a distinction between levels 0.25, 0.66 and 1.0 in this regard.

6.3 Evaluation of shared control

During the trials, the performance of the shared control system was evaluated from several angles. One of the most important of these, from a safety perspective, is the effect of the level of guidance on the forces that are experienced by the instrument tip in contact with the vessel walls. Specifically of interest here is the component of the force perpendicular to the tip, because this is the component that can cause a potential puncture. For this metric the projected tip force is measured in the direction of the final beam component of the instrument, as shown in eq. 6.1. Here, $\boldsymbol{t_{end}}$ is the line segment between the final and next to final elements of the beam model.

$$\boldsymbol{f}_{proj,tip} = \frac{\boldsymbol{f} \cdot \boldsymbol{t_{end}}}{|\boldsymbol{t_{end}}|} \tag{6.1}$$

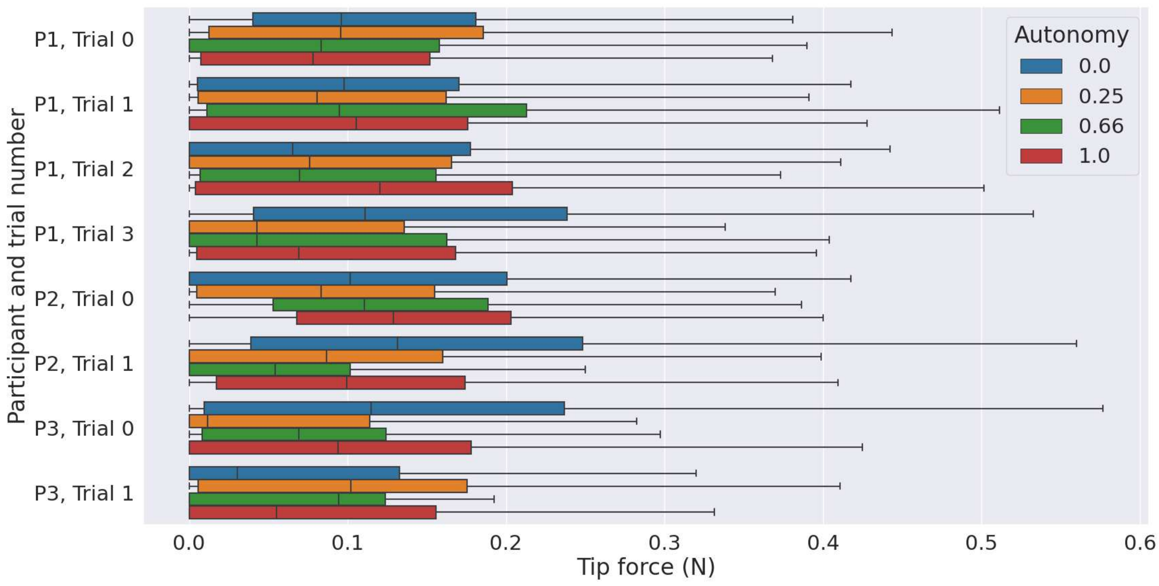

Figure 6.3: Box plots showing the quartiles and means (central line) of the distribution of the tip forces aligned with the instrument tip per participant/trial combination, sorted by guidance level. Outliers, defined as being beyond 1.5 times the interquartile range, are excluded from the figure.

Fig. ref 6.3 shows the distribution of forces over the combinations of participants and trials. As shown, there is significant spread within groups and individual trials. Because participants 1 and 3 were fairly novice to the system, with only experience in the earlier trials, it is possible there is a learning effect in the data. For this reason, fig. 6.4 shows the distribution of forces separately for each trial. It should be taken into account that only participant 1 did more than 2 trials. Nevertheless, the distribution shows that the mean of the data as well as the spread is fairly consistent across trial 0 and 1, and that there might even be a deterioration in performance for further trials. This can possibly be accounted to the participants being less careful with new hardware when they are more used to it. There is a difference between different participants, as shown on the left of fig. 6.4, with participant 3 outperforming the other two.

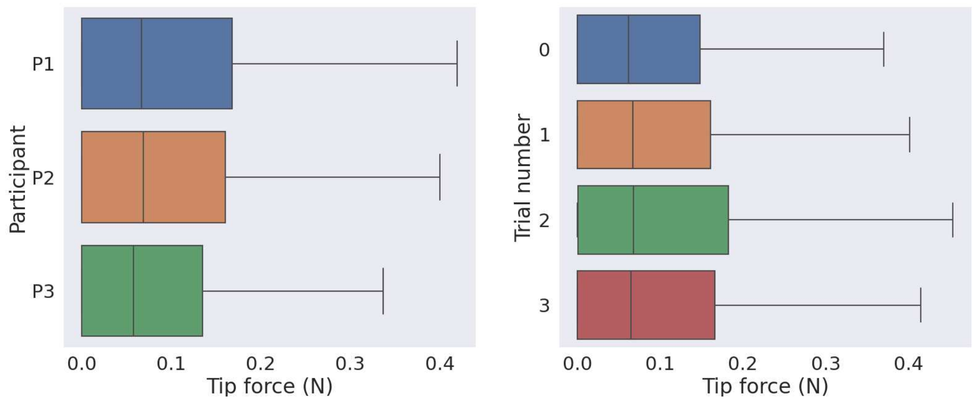

Figure 6.4: Box plots showing the quartiles and medians (central line) of the distribution of the tip forces aligned with the instrument tip per separate participant (left) and trial (right) over all guidance levels. Outliers, defined as being beyond 1.5 times the interquartile range, are excluded from the figure. Only non-zero forces are shown

Most significant to the subject of this thesis, however, is how the guidance level affects the results across participants and trial numbers. Figure 6.5 shows the spread for each level of guidance. The data seems to match well with the participants experience here, with the low (0.25) and medium (0.66) guidance levels outperforming trials with no or high (1.0) guidance on both mean and spread. Interesting to note is that the best-performing mean is the low level, while the data on the medium level is more closely distributed towards the low end.

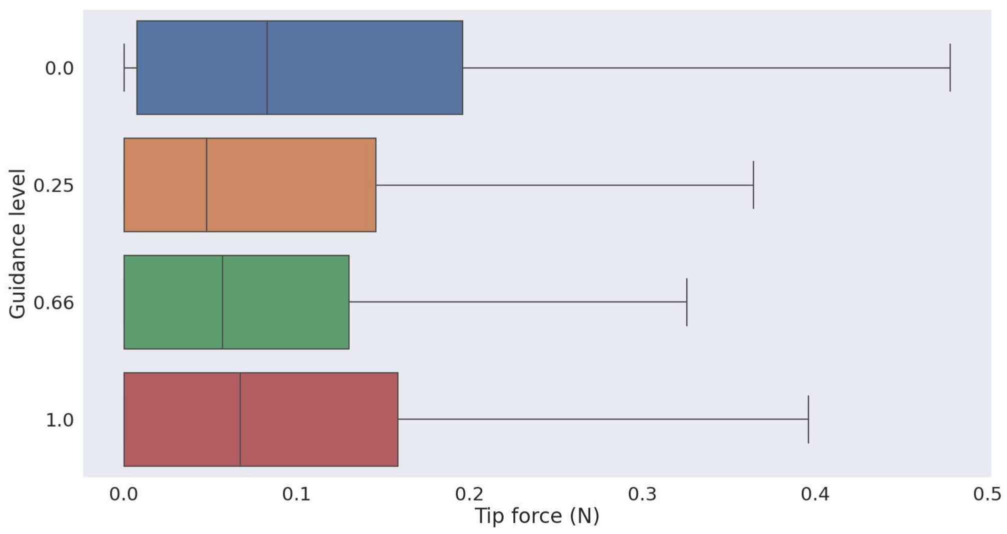

Figure 6.5: Box plots showing the quartiles and medians (central line) of the distribution of the tip forces per guidance level. Outliers, defined as being beyond 1.5 times the interquartile range, are excluded from the figure. Only non-zero forces are taken into account.

Table 6.3 shows more information on the outliers, which are impractical to show in figures due to the spread. None of the guidance levels is able to completely eradicate very high tip forces, but there is a visible effect on the high percentiles of the data. This effect is absent, or even negative, for the highest level of guidance.

Guidance level	**Mean**	**Median**	**q0.25**	**q0.75**	**q0.95**	**q0.99**	**q0.999**	**q1**
None (0)	0.137	0.083	0.007	0.196	0.487	1.721	4.520	12.26
Low (0.25)	0.117	0.047	0.000	0.146	0.413	1.687	2.360	3.131
Medium (0.66)	0.106	0.057	0.000	0.130	0.372	1.672	3.177	4.146
High (1.00)	0.147	0.067	0.000	0.158	0.768	2.492	6.352	16.77

Table 6.3: Tip force metrics per guidance levels. All numbers are in Newton (N). Columns starting with q indicate the start of that quantile, e.g.: q0.95 shows the threshold for the upper 5% of data for that guidance level

Additionally, a Kruskal-Wallis test was performed between the guidance levels to test if the distributions were likely to occur without variation in the independent variable. From this, we can conclude that the difference between guidance values is statistically significant (p $<$ 0.001). t-tests between sample mean and standard deviations of the different are shown in tab. 6.4. The test show a statistically significant reduction in the mean force for guidance levels 0.25 and 0.66.

	0.0	**0.25**	**0.66**	**1.0**
Mean $\pm$ std	0.137 $\pm$ 0.3155	0.117 $\pm$ 0.4460	0.106 $\pm$ 0.5451	0.147 $\pm$ 2.632
Sample size	37923	35367	42110	40463
p-value*	-	$<$0.001	$<$0.001	0.4482
95-CI-difference (abs)	-	[-0.0254, -0.0146]	[-0.0372, -0.0248]	[-0.016, 0.036]
95-CI-difference (% of 0.0 mean)	-	[-18.5, -10.7]	[-27.1, -18.1]	[-11.7, 26.3]

Table 6.4: t-test statistics on means of data grouped by guidance level compared to the no-guidance trials. All force data is in Newton.

Besides reduction in the interaction forces due to haptic feedback, it is also interesting to view the effect on the quantity of contacts. Fig. 6.6 shows the time spent in collisions ($\boldsymbol{f}_{tip,min} \geq 0.1$) as a ratio of overall active time as well as collision and haptic intervention rates per second of simulated time. Collision rates were determined by counting the number of points were the contact for moved sharply upwards from zero by thresholding the differential. A collision is counted when a minimum force, $\boldsymbol{f}$, is exceeded for a minimum of 5 consecutive time steps (125ms simulated time). Haptic intervention rates are determined similarly, with the haptic scaling exceeding 100% for at least 5 time steps. Note that as long as the forces or haptic feedback scaling stays above the threshold continuously, one instance is counted. On the other hand, even a momentary retraction will count as a new collision.

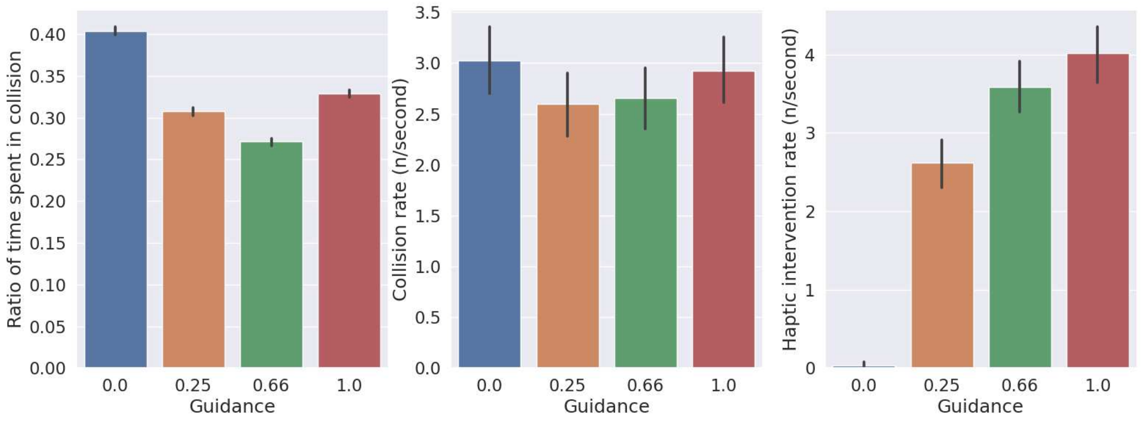

Figure 6.6: Statistics for collision occurrence against guidance levens. Left: Time spent in collision as a ratio of total active time. Middle: collision rate per second of simulated time. Right: haptic intervention rate per second of simulated time. In all figures, error bars show 95% confidence intervals between trials.

	0.0	**0.25**	**0.66**	**1.0**
Mean $\pm$ std	7.623 $\pm$ 6.532	6.294 $\pm$ 3.886	6.926 $\pm$ 3.776	6.702 $\pm$ 3.909
Sample size	37923	35367	42110	40463
p-value*	-	$<$0.001	$<$0.001	0.001
95-CI-difference (abs)	-	-1.4075 to -1.2505	-0.7701 to -0.6239	-0.9958 to -0.8462
95-CI-difference (% of 0.0 mean)	-	[-18.4, -16.4]	[-10.1, -8.1]	[-13.1, -11.1]

Table 6.5: t-test statistics on means of error norm data grouped by guidance level compared to the no-guidance trials. All distances are in mm.

The attitude controller is intended to steer the tip of the instruments towards the path, and should reduce the deviation between the tip and the path overall. As such, fig. 6.7 shows the distribution of path errors per guidance level. The error for this purpose is defined as the norm of the line of the final tip edge, p_{end} of the beam to the closest point on the path, p_{path}, as shown in eq. 6.2.

$$|e_{tip}| = |\boldsymbol{p}_{path} - \boldsymbol{p}_{end}| \tag{6.2}$$

In fig. 6.7 or any position data, no data is excluded as the force anomalies discussed before do not influence the trajectory of the instruments significantly. To distinguish the origin of outliers, the distribution per experiment is also shown. Noticeable at the bottom graph is that all three highest outliers clusters lie with the unguided systems, while the median error over the trajectory is consistent between autonomy levels, the top graph shows a trend over the outliers of the data. In table 6.5, a statistical t-test is performed on the error norms.

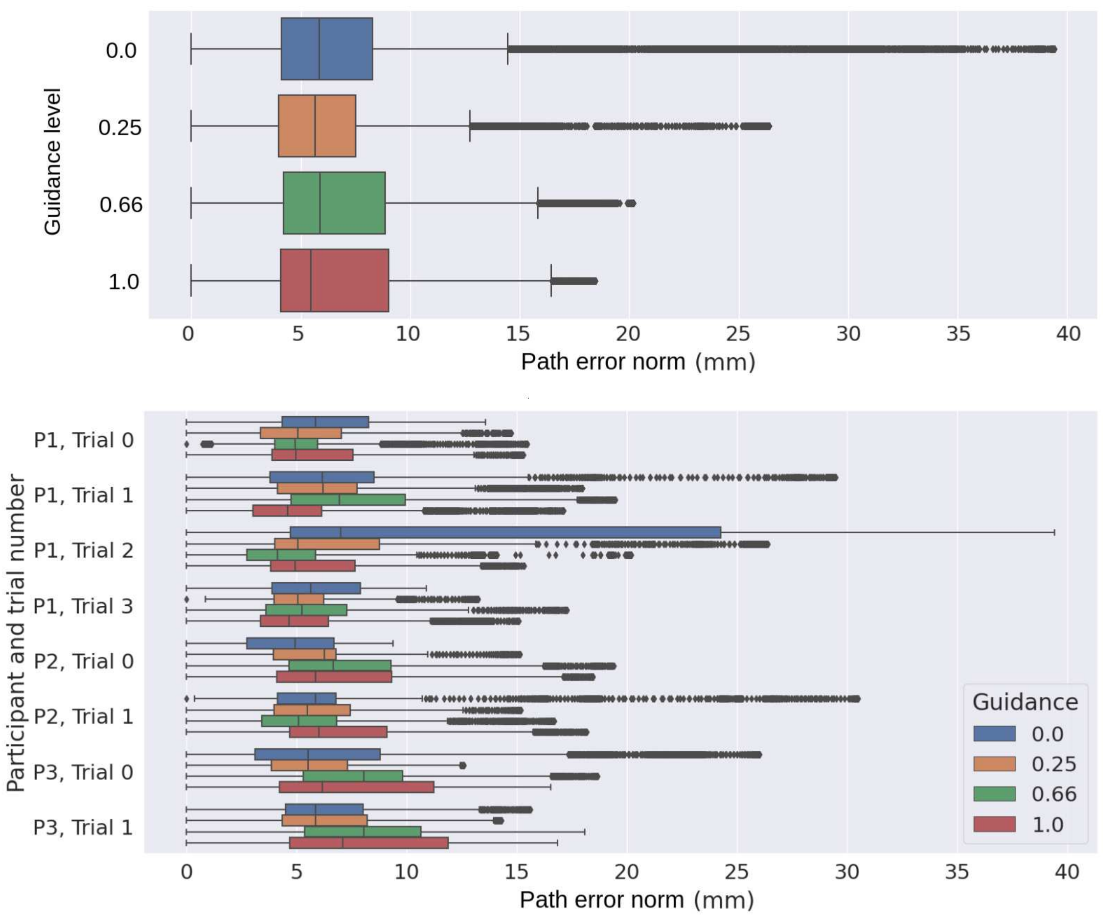

Figure 6.7: Distribution of error norms as defined in eq. 6.2 per guidance level (top), and per participant/trial combination (bottom).

Lastly, fig. 6.8 shows completion times in both simulated and real time grouped per experiment and as distributions per guidance level. Means and standard deviations are shown in tab 6.2 in the previous subsection.

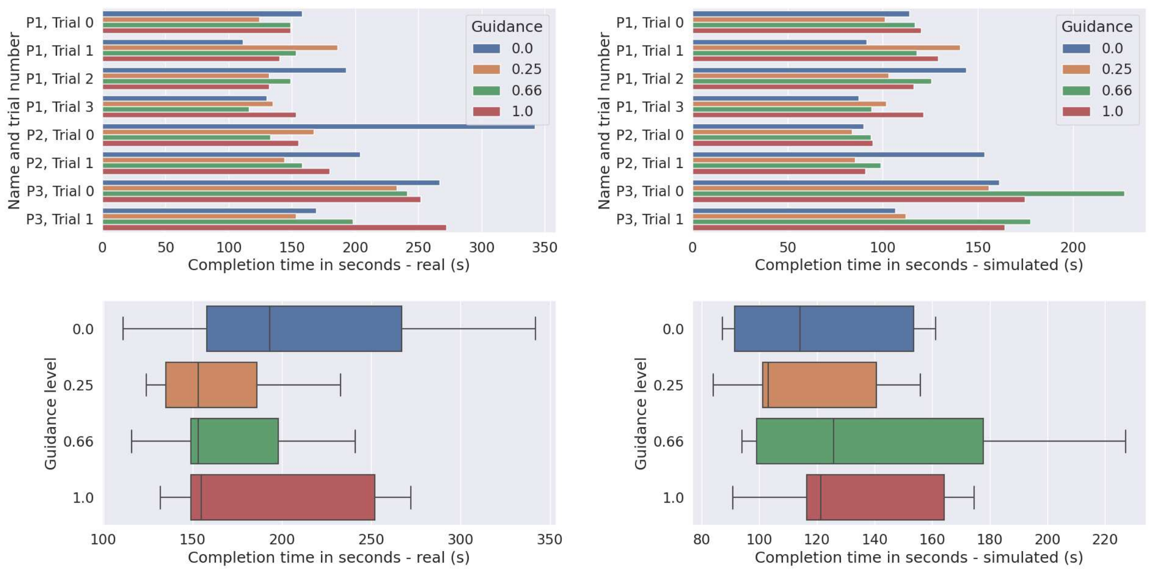

Figure 6.8: Completion times in simulated time and real time per participant, trial and autonomy level. Top: times per operation separated by participant, trial number and guidance level. Bottom: distributions of times per operation by guidance level. Left: completion times in real time, Right: completion times in simulated time. *Note*: P2, Trial 0 ran into a problem with the master device. The simulation was paused shortly and continued after.

Chapter 7

Discussion

In this discussion, the performance of both the framework and added controller are analyzed. In the previous chapter, it was found that a significant reduction in mean tip forces can be found by the addition of haptic feedback and attitude guidance at guidance level 0.25 and 0.66. The size of the effect, measured as the center of the CI-95 interval (see 6.3), is found to be an approximate reduction of 14.1% ($p < 0.001$) for the 0.25 level, and an approximate reduction of 22.5% ($p < 0.001$) for the 0.66 level in mean contact forces. No significant reduction or increase was found for guidance level 1.0.

Statistics on the error norms (tab. 6.5) likewise show a reduction in path errors for all non-zero guidance levels. The size of this effect seems greatest for the 0.25 guidance level, at an approximately 17.4% ($p < 0.001$) reduction from the 0.0 mean. A significant effect was also observed at the 0.66 and 1.0 guidance levels of 9.1% ($p < 0.001$) and 12.1% ($p = 0.001$), respectively.

Additionally, it was found that the users that performed guided operations spent overall less time in collision, with possibly slightly reduced rates of collision. This does not meet standards for academic significance, however, and will need further investigation. Similarly, no correlation was found between the completion times, both in simulation and real time, and the guidance levels.

On the side of technical performance, the framework was found to perform its primary functions during the trials. Computation performance was found to average at rates of $\tilde{2}8.8$ frames per second for 0.719 simulated seconds per second of real time (tab. 6.2) with significant caveats related to the stage of the trial and the amount of colliding objects (tab. 6.1). No significant malfunctions were noted during the trials with regard to haptic feedback, visual feedback and the communication pipeline. Some technical issues surfaced with regard to the force calculation and master hardware: improvements are necessary to improve reliability of user inputs and prevent anomalous forces from occurring during instrument changes, but the framework can overall be concluded to fulfill its primary purpose.

7.1 Contact force reduction

The contact force reductions for guidance levels of 0.25 and 0.66 with 14.1% ($p < 0.001$) and 22.5% ($p < 0.001$) respectively match well with the goal of the framework as stated in the introduction, sec. 1.2 as well as the subjective experience of the tested participants. Participants noted that they felt most comfortable with the low and medium levels, which is understandable as these still allow the use to override the control input with relative ease. As the practical path of surgical instruments will not always follow a central curve or neat preset trajectory, it is essential that operators are allowed the ability to intervene in the operation. Further improvements in this reduction will be necessary to enhance the provided possible safety improvement. Similar works, [47] have reductions in similar scale with fully automated works, but generally with lower sample sizes, completion rates and higher forces overall. Other researchers, such as in [23], have achieved larger reductions with machine-learning-based systems, but are unclear in whether further improvements and scaling are possible. In any case, the shown reduction provides some confidence in the approach in general, and provides a starting point for further development of the platform and research into its capabilities for contact force mitigation.

Although a good p-value does indicate that results are unlikely to be caused by chance, there are caveats to be placed with the results in general. When inspecting the variation of force distributions per participant and trial number, such as in 6.4, some variation is visible as well. Unfortunately, the inclusion of only three participants makes it very difficult to accurately address the variability of the data in this regard, but fig. 6.3 does hint that there is a trend with participants as well, with the highest three outliers all being unguided trials. Furthermore, 6 out of 8 trials sets report the lowest median force with either the low or medium guidance levels. Of course, further investigation will be necessary to confirm the presence within trial and participant groups and confirm the presence of an effect on this metric. As such, it is recommended that the trials are repeated in the future with a significantly higher number of participants.

Besides the number of participants, there is another dimension where the current results fail to confirm efficacy: experience. As endovascular operations are generally not performed by beginners, any framework meant to eventually be used for endovascular surgery should be tested on the surgeons themselves. As few of these were unfortunately available, the tests were performed with surgical novices. Some participants did have some earlier experience with the master device and framework, but this dimension was purposefully left out of the initial analysis as the differences are hard to quantify. Additionally, it can be seen in both fig. 6.4 and fig. 6.3 that inter-trial variation is as great or greater than inter-participant variation here. Nevertheless, the absence of true experts leaves open the question if a shared controller would be able to (partially) mitigate any experience difference.

Besides the magnitudes of contact, there is reason to further investigate the number of collisions and time spent in these collision on base of the data as well. While these effects were not the main target of investigation for this work, it seems sensible that an attitude controller might reduce the amount of contacts in two ways: by preventing collisions that the operator would have made by accident in an unguided section, or by contra-productively lengthening collision moments the operator would have tried to retreat from earlier. It is therefore important to compare these values with the time spent in collision or, in future work, time spent in each collision. That both the time spent in collision and the measure of collision rate show some reduction for all levels of guidance forms another indication that the shared controller, once further developed and thoroughly tested, might have a potentially positive effect on the prevention of vessel wall damage during endovascular surgery.

Interestingly, data recorded on the amount of haptic interventions gives an indication that the amount of interventions increases strongly with the guidance level. It is hypothesized that the reason for this increase might be that a very high haptic scaling can somewhat destabilize the virtual mass-damper in the master device, and cause oscillations that are taken in as position commands, further increasing the amount of haptic interventions when near a collision point. This too, however, will need further investigation by performing a closer study on the inputs provided by the master combined with video evidence of the user operating it.

In summary, this work provides positive results for low and medium guidance levels with a strong indication that mean contact forces can be reduced, and some evidence that the quantity of contact moments can be reduced with presented combined approach of shared attitude control and haptic feedback. While the absence of significant improvements for the high guidance level do not disprove the utility of complete autonomous devices in any way, it does carefully speak for the viability of keeping even novice human operators involved in the control loop. Future research will need to prove if this approach is truly optimal, and if the combination of especially expert skill with a more optimized control algorithm can be greater than the sum of its parts.

7.2 Effects on navigation

Besides the effects of the implemented shared control method on contact forces, an investigation was also performed on the navigational effects of the shared control. A significant reduction ($p < 0.001$) was found on the error norms with respect to the calculated path via implementation of the controller, with the highest reduction for the low (0.25) level guidance implementation with a CI-95 interval centered at a reduction of 17.4% (see tab. 6.5). All levels of shared control were found to have a significant impact in this regard.

Of course, it makes sense that a controller designed to align the tip towards the path direction might reduce errors in relation to that path, but this does not automatically provide motion along the path, just an alignment towards the path. Essentially, this alignment is meant to guide the instruments in the right direction during a cornering maneuver, where the greatest errors would be expected to exist. It is therefore an indication that the controller works as intended that it is able to reduce the mean error.

In addition to the mean error which gives a measure of the trajectory-wide performance, the controller is especially meant to interfere during mishaps. Mistakes are difficult to quantify exactly due to their rarity and inconsistency, but a strong indication of a reduction in outlier errors can be seen with respect to the unguided trials in fig. 6.7 (bottom). Interestingly, the most notable outliers are all found with the unguided trials, and don't seem to have happened during the first trial series.

Lastly, completion time is often cited in literature as a significant aim in autonomous device (e.g., see [26] [27]), and the results in this regard are shown in this work as well in fig. 6.8. Both the real and simulated completion times vary wildly. Some clustering can be seen around the minimum times, as there are several performances where the instruments move near maximum speeds and nothing significant goes wrong during the trial. In both cases, there are outliers that more than double the time spent, and these seem to occur for all levels of guidance.

Overall, effects on the navigation can be concluded to be present but modest in size in regard to the mean error. Future work will need to investigate more closely if a reduction in large errors (outliers) is present.

7.3 Framework

With respect to the performance of the framework, several improvements can be suggested. The most obvious improvement relates to computational performance. While mean frame rates however around 35 frames per second, a severe reduction in performance is seen past the second corner of the mesh, as detailed in tab. 6.1. This reduction in performance is caused by the severe increase in collision calculations. Only because steps have already been undertaken to reduce these by reducing mesh complexity, increasing solver tolerances and reducing maximum solver iterations during development, can interactive simulation rates be achieved. It is hypothesized that a great improvement can be obtained by optimizing the collision pipeline for either reliable multi-processing or the use of GPU for computations. It is unfortunate that this is currently not possible in SOFA, either in the stock releases or with help of the plugins dedicated to this purpose. The nature of the long critical path of the collision pipeline unfortunately means that optimization might be complex, and that this might not be something that is achieved in the very near future.

Another specific avenue for future improvement is the nature of the visual feedback. If the framework will ever be tested on experienced professionals, it will need a significant amount of visual improvements to increase perceived realism. Of specific important are scaling of forces and especially realism of the anatomy.

The results in the previous sections also allow for an evaluation of the forces as rendered in SOFA. With an average contact force in the 0.1N range, it matches well with previous work on in-vivo anatomies [46]. The mean force is comparable in size to the moderate force listed, and the strong force listed matches with the upper 15-20% of measures forces. This does however not hold for the extreme outliers recorded in table 6.3. The maximum forces recorded in the simulator can exceed 10N in extreme circumstances (for the 0.0 and 1.0 guidance levels). While this matches the most extreme forces in [46], the author lists 7-to-1 odds on a puncture with this force. It is therefore recommended that further research is done on the origin of these forces to whether they are realistic in relation to the interaction that causes them, and how they can be prevented.

While the framework performs its basic functions well, a dedicated (group of) researcher(s) could find several options to improve it. While the shared control implementation was tested in a generated mesh that has similar characteristics to human blood vessels, this is not a realistic anatomical representation of a human patient. Luckily, a large part of the meshing pipeline is already in place to convert medical imaging files to STL meshes, as these commonly use the same .mhd/raw format the VascuSynth generator does. While the importing of anatomical data might be in place, there is no guarantee that the control implementation will scale well with a real anatomy. It is also for this reason that this is such a necessary direction for future research: real validation of the approach will require a more comprehensive, detailed simulator and likely refinement of communication, control and testing methods.

Chapter 8

Conclusion

This work presents a working framework for interactive simulation of endovascular procedures using a robotic master controller in the simulation loop. The framework is concluded to be functional in the primary simulation functions, and able to convincingly calculate the collision forces during endovascular operations. A combined implementation of shared attitude control, haptic and visual feedback is provided in the work with the aim to mitigate contact forces on the anatomy. The shared attitude controller and haptic feedback scale with a guidance parameter that determines the level of interventionism the controller applies. User trials with four levels of guidance were conducted, from which a significant reduction in mean forces is concluded. Additionally, some positive effects on the mean path error were found, as well as indications of positive effects on the amounts of contacts and time spent in contact. The framework can be concluded to be effective in this early stage of development, and provides a base for further development and testing.

Bibliography

[1] R. Assi and A. Dardik, “Endovascular training of vascular surgeons in the usa.” *Annals of Vascular Diseases*, vol. 5, no. 4, pp. 423–7, 2012.

[2] A. K. Thukkani and S. Kinlay, “Endovascular intervention for peripheral artery disease,” *Circulation research*, vol. 116, p. 1599–1613, 2015.

[3] D. Kundrat, G. Dagnino, T. M. Y. Kwok, M. E. M. K. Abdelaziz, W. Chi, A. Nguyen, C. Riga, and G.-Z. Yang, “An mr-safe endovascular robotic platform: Design, control, and ex-vivo evaluation,” *IEEE Transactions on Biomedical Engineering*, vol. 68, no. 10, pp. 3110–3121, 2021.

[4] P. Legeza, G. W. Britz, T. Loh, and A. Lumsden, “Current utilization and future directions of robotic-assisted endovascular surgery,” *Expert Review of Medical Devices*, vol. 17, no. 9, pp. 919–927, 2020, pMID: 32835546. [Online]. Available: https://doi.org/10.1080/17434440.2020.1814742

[5] G. Dagnino, D. Kundrat, T. M. Y. Kwok, M. E. M. K. Abdelaziz, W. Chi, A. Nguyen, C. Riga, and G.-Z. Yang, “In-vivo validation of a novel robotic platform for endovascular intervention,” *IEEE Transactions on Biomedical Engineering*, pp. 1–9, 2022.

[6] L. Cruddas, G. Martin, and C. Riga, “Robotic endovascular surgery: current and future practice,” *Seminars in Vascular Surgery*, vol. 34, no. 4, pp. 233–240, 2021.

[7] Y. Kassahun, B. Yu, A. Tibebu *et al.*, “Surgical robotics beyound enahnced dexterity instrumentation: a survey of machine learning techniques and their role in intelligent and autonomous surgical actions,” *International Journal of Computer Assisted Radiology and Surgery*, vol. 11, pp. 553–568, 2016.

[8] Y. Zhao, S. Guo, Y. Wang *et al.*, “A cnn-based prototype method of unstructured surgical state perception and navigation for an endovascular surgery robot,” *Medical & Biological Engineering & Computing*, vol. 57, pp. 1875–1887, 2019.

[9] T. Behr, T. P. Pusch, M. Siegfarth, D. Hüsener, T. Mörschel, and L. Karstensen, “Deep reinforcement learning for the navigation of neurovascular catheters,” *Current Directions in Biomedical Engineering*, vol. 5, no. 1, pp. 5–8, 2019.

[10] J. Kweon, K. Kim *et al.*, “Deep reinforcement learning for guidewire navigation in coronary artery phantom,” *IEEE Access*, vol. 9, pp. 166 409–166 422, 2021.

[11] H. You, E. Bae, Y. Moon, J. Kweon, and J. Choi, “Automatic control of cardiac ablation catheter with deep reinforcement learning method,” *Journal of Mechanical Science and Technology*, vol. 33, no. 11, p. 5415 – 5423, 2019.

[12] B. Murali, V. Belvroy, S. Pandey, M. Byrne, J. Bismuth, and M. O’Malley, “Towards automated performance assessment using velocity-based motion quality metrics,” in *International Symposium on Medical Robotics*, 11 2020, pp. 36–42.

[13] J. Allard, S. Cotin, F. Faure, P.-J. Bensoussan, F. Poyer, C. Duriez, H. Delingette, and L. Grisoni, “SOFA - an Open Source Framework for Medical Simulation,” in *MMVR 15 - Medicine Meets Virtual Reality*, ser. Studies in Health Technology and Informatics, Feb. 2007, vol. 125, pp. 13–18.

[14] S. Consortium, “Sofa beamadapter plugin,” Dec 2021, version 22.12. [Online]. Available: https://github.com/sofa-framework/BeamAdapter

[15] O. M. O’Reilly, *Kirchhoff’s Rod Theory*. Cham: Springer International Publishing, 2017, pp. 187–268. [Online]. Available: https://doi.org/10.1007/978-3-319-50598-5_5

[16] E. Cosserat, F. Cosserat, M. Brocato, and K. Chatzis, *Theorie des Corps Deformables*. Hermann et fils, 1909.

[17] C. Duriez, S. Cotin, J. Lenoir, and P. Neumann, “New approaches to catheter navigation for interventional radiology simulation,” *Computer Aided Surgery*, vol. 11, no. 6, pp. 300–308, 2006, pMID: 17458764. [Online]. Available: https://doi.org/10.3109/10929080601090623

[18] J. Przemieniecki, “Matrix structural analysis of substructures,” *AIAA Journal*, vol. 1, no. 1, pp. 138–147, 1963.

[19] T. G. Kolda and B. W. Bader, “Tensor decompositions and applications,” *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[20] A. Witkin, “Physically based modeling: Principles and practice - constrained dynamics,” *Robotics Institute, Carnegie Mellon University*, pp. 4–11, 1997. [Online]. Available: https://www.cs.cmu.edu/~baraff/sigcourse/notesf.pdf

[21] C. Duriez, “Real-time haptic simulation of medical procedures involving deformations and device-tissue interactions,” Ph.D. dissertation, Université des Sciences et Technologie de Lille - Lille I, 2013.

[22] C. Guébert, C. Duriez, and L. Grisoni, “Unified processing of constraints for interactive simulation,” in *Workshop in Virtual Reality Interactions and Physical Simulation VRIPHYS’ 2008*. Eurographics association, 2013. [Online]. Available: https://hal.science/hal-00823764

[23] W. Chi, J. Liu, M. E. M. K. Abdelaziz, G. Dagnino, C. Riga, C. Bicknell, and G.-Z. Yang, “Trajectory optimization of robot-assisted endovascular catheterization with reinforcement learning,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3875–3881.

[24] Y. Zhao, Y. Wang, J. Zhang, X. Liu, Y. Li, S. Guo, X. Yang, and S. Hong, “Surgical gan: Towards real-time path planning for passive flexible tools in endovascular surgeries,” *Neurocomputing*, vol. 500, pp. 567–580, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231222006038

[25] L. Karstensen, T. Behr, T. P. Pusch, F. Mathis-Ullrich, and J. Stallkamp, “Autonomous guidewire navigation in a two dimensional vascular phantom,” *Current Directions in Biomedical Engineering*, vol. 6, no. 1, p. 20200007, 2020. [Online]. Available: https://doi.org/10.1515/cdbme-2020-0007

[26] Z. Li, J. Dankelman, and E. De Momi, “Path planning for endovascular catheterization under curvature constraints via two-phase searching approach,” *International Journal of Computer Assisted Radiology and Surgery*, vol. 16, no. 4, pp. 619–627, Apr 2021. [Online]. Available: https://doi.org/10.1007/s11548-021-02328-x

[27] J. Fauser, M. Fuchs *et al.*, “Preoperative planning for guidewires employing shape-regularized segmentation and optimized trajectories,” in *OR 2.0 Context-Aware Operating Theaters and Machine Learning in Clinical Neuroimaging*, 2019, pp. 12–20.

[28] Y. Ganji, F. Janabi-Sharifi *et al.*, “Robot-assisted catheter manipulation for intracardiac navigation,” *International Journal of Computer Assisted Radiology and Surgery*, vol. 4, no. 4, pp. 307–315, 2009.

[29] A. Shademan, R. S. Decker, J. D. Opfermann, S. Leonard, A. Krieger, and P. C. W. Kim, “Supervised autonomous robotic soft tissue surgery,” *Science Translational Medicine*, vol. 8, no. 337, pp. 337ra64–337ra64, 2016.

[30] A. A. Jamjoom, A. M. Jamjoom *et al.*, “Autonomous surgical robotic systems and the liability dilemma,” *Frontiers in Surgery*, vol. 9, 2022.

[31] M. Selvaggio, A. M. G. E, R. Moccia, F. Ficuciello, and B. Siciliano, “Haptic-guided shared control for needle grasping optimization in minimally invasive robotic surgery,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 3617–3623.

[32] L. Xiong, C. Chng *et al.*, “Shared control of a medical robot with haptic guidance,” *International Journal of Computer Assisted Radiology and Surgery*, vol. 12, pp. 137–147, 2017.

[33] A. Dragan and S. Srinivasa, “A policy-blending formalism for shared control,” *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 790–805, 2013.

[34] J.-w. Choi, R. Curry, and G. Elkaim, “Path planning based on bézier curve for autonomous ground vehicles,” in *Advances in Electrical and Electronics Engineering - IAENG Special Edition of the World Congress on Engineering and Computer Science 2008*, 2008, pp. 158–166.

[35] M. Kamermans, “A primer on bezier curves,” 2011. [Online]. Available: https://pomax.github.io/bezierinfo/

[36] D. L. Finn, “Ma 323 geometric modelling course notes: Day 18 bezier splines ii,” January 2005.

[37] P. Jassi and G. Hamarneh, “Vascusynth: Vascular tree synthesis software,” *Insight Journal*, vol. January-June, pp. 1–12, 2011.

[38] F. Divo, B. Powell *et al.*, “Python-can module.” [Online]. Available: https://python-can.readthedocs.io/

[39] M. E. M. K. Abdelaziz, D. Kundrat, M. Pupillo, G. Dagnino, T. M. Y. Kwok, W. Chi, V. Groenhuis, F. J. Siepel, C. Riga, S. Stramigioli, and G.-Z. Yang, “Toward a versatile robotic platform for fluoroscopy and mri-guided endovascular interventions: A pre-clinical study,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 5411–5418.

[40] “Matweb: Material property data,” https://www.matweb.com, accessed: 2023-01-30.

[41] M. McCormick, X. Liu, J. Jomier, C. Marion, and L. Ibanez, “Itk: enabling reproducible research and open science,” *Frontiers in Neurology*, vol. 8, no. 13, 2014.

[42] O. Au, C. Tai, H. Chu, and T. Lee, “Skeleton extraction by mesh contraction,” *ACM Transactions on Graphics*, vol. 27, no. 3, p. 44, 2008.

[43] “Rosetta code: Ray-casting algorithm,” https://rosettacode.org/wiki/Ray-casting_algorithm, accessed: 2022-12-19.

[44] T. Rizk, D. Patel, N. Dimitri *et al.*, “Iatrogenic arterial perforation during endovascular interventions,” *Cureus*, vol. 12, no. 6, 2020.

[45] H. Hlaing and A. Sin, “Admittance controller for physical human-robot interaction using one-dof assist device,” *International Journal of Scientific and Engineering Research*, vol. 8, no. 12, 2017.

[46] Y. Okumura, S. Johnson *et al.*, “A systematical analysis of in vivo contact forces on virtual catheter tip/tissue surface contact during cardiac mapping and intervention,” *Journal of Cardiovascular Electrophysiology*, vol. 19, pp. 632–640, 2008.

[47] W. Chi, G. Dagnino, T. Kwok, A. Nguyen, D. Kundrat, M. Abdelaziz, C. Riga, C. Bicknell, and G. zhong Yang, “Collaborative robot-assisted endovascular catheterization with generative adversarial imitation learning,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2414–2420.

Appendix A

Instrument properties

Property	**Catheter**	**Guidewire**
Type	Beacon Tip 5 Fr VanSchie2	Radifocus^{TM} Guide Wire M .035” 180 cm angled
Length	100.0 cm	100.0 cm
Tip length	100.0 mm	30.0 mm
Material	Braided nylon	Nitinol
Young modulus (tip)	3 GPa (3 GPa)	75 GPa (40 GPa)
Internal radius	0.885 mm (5 Fr$*$)	0.440 mm
Curvature diameter	190.0 mm	30.0 mm
Simulation parameters		
Number of edges	200	200
Number of edges in collision model (tip)	40 (20)	50 (10)
Density of beams (tip)	40 (10)	30 (5)
Contact distance	2.0 mm	1.0 mm

Table A.1: Material and simulation properties of instruments

Appendix B

Software dependencies

The following dependencies are necessary to run the delivered software package:

Name	**URL**	**Tested version**
SOFA Framework	source	v22.12
SOFA BeamAdapter plugin	source	v22.12
SOFAPython3 plugin	Installed via SOFA plugin manager	v3.0
Python	source	3.10.11
python-can	source	v4.2.0
IXXAT CAN2USB driver	source	2.0.378

Additionally, all the necessary python modules listed as imports must be installed. They can all be installed via `pip` except `python-can`

Appendix C

User instructions

The software package that is delivered with this thesis is set up at the experimental PC, and can be used there. Setting up the can communication requires a bash script that can be run with the following command:

```
> ./Documents/can_setup/canstart.sh
```

Listing C.1: Linux terminal commands

This bash script reinstalls the IXXAT driver and activates the can installation. The reinstallation of the driver upon start up is an unfortunate necessity at this stage of development to run communications. After the reinstallation, the following commands can be used to test to use the CAN pipeline for debugging purposes:

```
> cansend can0 [ID][hashtag][MESSAGE]
> candump can0
```

Listing C.2: Linux terminal commands.

The commands enable the sending of messages and the dumping of all received communication in the terminal window, respectively. Note that `[hashtag]` should be replaced with ”#”

A callibration sequence must be run on the master before use. When it is powered, the user must move the handle from one end of the slotted rod to the other. After that, an acknowledgement is send via the terminal: `> cansend can0 032#0000000000000000`. To launch the package, the script `simulate.py` is run. The user enters their name, desired guidance levels, trial number and desired endpoint. For a reference of endpoint number, the script `pathing.py` can be run and changing the arguments of the `P.plot_bezier_with_points(...)` line to:

```
    P.plot_bezier_with_points(input_list, cp_list, cp_spline=cp_spline
   , plot_mesh=True, plot_skeleton=False, plot_C=True, labels=True)
```

Listing C.3: plotting node numbers on the mesh