Università degli Studi di Modena e Reggio Emilia

DOTTORATO DI RICERCA IN
INGEGNERIA DELL'INNOVAZIONE INDUSTRIALE

Ciclo XXVII

Interaction Control for Autonomous Robotic Surgery

Candidata: **Federica Ferraguti**

Direttore della Scuola:
Prof. Mauro Dell'Amico

Relatore:
Prof. Cristian Secchi

Esame finale anno 2015

Copyright ©2015 by Federica Ferraguti.

Reggio Emilia, Italy, March 2015.

Abstract

The introduction of robotic surgery within the operating rooms has significantly improved the quality of many surgical procedures. Recently, the research on medical robotics systems focused on increasing the level of autonomy in order to give them the possibility to carry out simple surgical actions in an autonomous way. However significant technological steps have to take place before even the smallest autonomous task is ready to be presented to the regulatory authorities.

This thesis addresses the initial steps of this process, in particular the development of control concepts satisfying the basic safety requirements of robotic surgery, i.e. providing the robot with the necessary dexterity and a stable and smooth behavior of the surgical tool. Three specific situations are considered: the automatic adaptation to variable tissue stiffness, the transition from autonomous to teleoperated mode and teleoperation of a dual-arm robotic manipulator. These situations replicate real life cases when the surgeon adapts the stiffness of her/his arm to deal with tissues of different consistency and when, due to unexpected events, the surgeon has to take over the control of the surgical robot. To address the first case, a passivity-based interaction control architecture that allows to implement safe and stable time-varying interaction behaviors is proposed. For the second case, a two-layered bilateral control architecture is introduced to ensure a safe behavior during the transition between autonomy and teleoperation and to still retain high performance control after the switch. Finally, to address the third case, a task-oriented teleoperation strategy is proposed.

The proposed strategies are integrated using a component-based software architecture to control a novel robot specifically designed for surgical applications during the FP7 European funded project Intelligent Surgical Robotics (I-SUR). Modularity and reconfigurability features of the robot automation system are highlighted through the implementation of some interesting use case scenarios: needle insertion and wound suturing.

Sommario

L'introduzione della robotica in ambito medico ha migliorato in modo significativo la qualità di molte operazioni chirurgiche. La ricerca sui sistemi di chirurgia robotica si è focalizzata recentemente sull'incremento del livello di autonomia del sistema, in modo che alcune semplici operazioni chirurgiche possano essere effettuate autonomamente. Tuttavia, sono necessari ancora notevoli sforzi tecnologici prima che anche la più piccola operazione autonoma possa essere autorizzata dalle autorità competenti.

Questa tesi si focalizza sulle fasi iniziali di questo processo, in particolare sullo sviluppo di algoritmi di controllo che soddisfino i requisiti di base della chirurgia robotica: fornire al robot la destrezza necessaria ad effettuare la procedura e garantire che il movimento dello strumento chirurgico sia stabile e sicuro. In particolare, vengono prese in considerazione tre specifiche situazioni: l'adattamento automatico del comportamento del robot durante l'interazione con tessuti di diversa rigidezza, il passaggio dalla modalità automatica alla modalità teleoperata e la teleoperazione di un manipolatore a due braccia. Questi eventi ricalcano situazioni di vita reale, in cui un chirurgo, nell'effettuare un'operazione, deve adattare la rigidezza del suo braccio quando interagisce con tessuti di diversa consistenza oppure in cui, a causa di un evento inaspettato, il chirurgo deve poter prendere il controllo del robot e passare quindi in teleoperazione. Per risolvere la prima problematica viene proposta un'architettura di controllo, basata sul concetto di passività, che permette di implementare comportamenti interattivi con parametri variabili nel tempo mantenendo la stabilità del sistema e quindi la sicurezza. Nel secondo caso, viene introdotta un'architettura di controllo bilaterale a due livelli che garantisce una transizione sicura da modalità automatica a modalità teleoperata mantenendo elevate prestazioni. Infine, per risolvere la terza problematica, viene proposta una strategia di teleoperazione le cui caratteristiche variano a seconda del task da svolgere.

Le strategie proposte sono state integrate mediante un'architettura soft-

ware modulare e sono state utilizzate per controllare un nuovo robot progettato e costruito appositamente per applicazioni chirurgiche nel corso del progetto europeo Intelligent Surgical Robotics (I-SUR). Le caratteristiche di modularità e riconfigurabilità del sistema sviluppato sono validate attraverso l'implementazione di alcuni interessanti casi d'uso: l'inserimento di aghi e la sutura di ferite.

Acknowledgements

This thesis is the result of several years of work, and I really need to thank who made it possible.

I would like to express my deepest thanks to my supervisor, Prof. Cristian Secchi, and to Prof. Cesare Fantuzzi for giving me this opportunity and for their help, teachings and support during these years.

Many thanks go to Dr. Marcello Bonfè for his assistance and guidance in getting my Ph.D. He was a mentor, but then he became even a great friend.

I'd like to thank Prof. Dr. Roger Gassert and Dr. Olivier Lambercy for their great support during my four months at the Rehabilitation Engineering Lab, ETH - Zurich. It was a crucial and stressful period for our project and they really helped me in approaching everything in the right way.

I'd also like to thank all the members of the ARSControl group. Our days in the laboratory environment were full of humor and entertainment and this allowed me to always work with the right soul.

Sincere thanks also go to Silvana and Peter Egloff for their hospitality during my months in Zurich. Sometimes it is difficult to leave everything to go abroad, even if for only four months. I was really lucky in finding a second family there.

I really need to thank my parents for their support during these years. Their help and presence was an important constant, inside the ups and down of my life. Thanks for everything.

And, of course, my greatest thanks go to Nicola. Thanks for being there, always.

Contents

Chapter 1

Introduction

The introduction of minimally-invasive surgery (MIS) first and, more recently, of surgical robots, has brought new perspective to surgery and has significantly improved the quality of many critical surgical procedures. The advantages of minimally invasive surgery are very popular among surgeons, patients, and insurance companies. Indeed, by using minimally invasive surgery techniques many benefits can be offered to the patient over traditional surgeries:

- **Less pain.** MIS procedures cause less post-operative pain and discomfort. Studies have shown that patients undergoing MIS procedures report less pain and require smaller doses of pain relievers than patients undergoing traditional surgeries.

- **Shorter hospital stay.** Shorter hospital stay and quicker return to normal activities. Patients who undergo MIS procedures are usually able to go home sooner. And, in many cases, the patient is able to return to normal activities and work more quickly.

- **Less scarring.** MIS procedures require smaller incisions, which means smaller, less noticeable scars. The scars that do form as a result of MIS typically have a less jagged edge, giving them a more appealing look.

- **Less injury to tissue.** Most traditional surgeries require a long incision. This incision usually has to be made through muscle. Muscle needs a significant time to heal after surgery. Because there are no long incisions in MIS, surgeons often do not have to cut through muscles to complete the procedure, leading to less tissue damage and quicker recovery.

- **Higher accuracy rate.** A higher accuracy rate for most procedures. Because MIS procedures use video-assisted equipment, the surgeon has better visualization and

magnification of internal organs and structure. For patients, this translates into a more accurate and definitive procedure.

As attractive as minimally-invasive surgery is, there are several limitations [1]. Some of the more prominent limitations involve the technical and mechanical nature of the equipment. Inherent in current minimally-invasive equipment is a loss of haptic feedback (force and tactile), natural hand-eye coordination, and dexterity. Moving the instruments while watching a 2-dimensional video monitor is somewhat counter-intuitive. One must move the instrument in the opposite direction from the desired target on the monitor to interact with the site of interest. Hand-eye coordination is therefore compromised (*fulcrum effect*). Moreover, current instruments have restricted degrees of motion; most of them have 4 degrees of motion, whereas the human wrist and hand have 7 degrees of motion. There is also a decreased sense of touch that makes tissue manipulation more heavily dependent on visualization. Finally, physiologic tremors in the surgeon are readily transmitted through the length of rigid instruments.

In recent years, surgical robots that help to overcome the limitations of minimally-invasive surgery and enhance the capabilities of the surgeon have been developed. The history of robotics in surgery begins with the Puma 560, a robot used in 1985 to perform neurosurgical biopsies with greater precision [2]. Three years later, a transurethral resection of the prostate was performed using the Puma 560 [3]. This system eventually led to the development of Probot [4], a robot designed specifically for transurethral resection of the prostate. While Probot was being developed, a robotic system capable of interventions on rigid tissue (i.e. bones drilling or cutting) was developed: the ROBODOC system [5]. ROBODOC was the first surgical robot approved by the United States Food and Drug Administration (FDA) [6].

Today, in addition to ROBODOC, several other robotic systems have been commercially developed and approved by the FDA for general surgical use. These include the comprehensive master-slave surgical robotic systems Da Vinci by Intuitive Surgical [7] and Zeus by Computer Motion.

The Da Vinci and Zeus systems are similar in their capabilities but different in their approaches to robotic surgery. Both systems are comprehensive master-slave surgical robots with multiple arms operated remotely from a console with video assisted visualization and computer enhancement.

In the Da Vinci system (Fig. 1.1) there are essentially 3 components:

- A *vision cart* that holds a dual light source and dual 3-chip cameras. The camera arm contains dual cameras and the image generated is 3-dimensional.

Figure 1.1: The Da Vinci surgical system with its surgeon's consol (left) and the robot arms (right).

Figure 1.2: The Zeus surgical system with its table-mounted arms (left) and the surgeon's console (right).

- A *master console* where the operating surgeon sits. It consists of: an image processing computer that generates a true 3-dimensional image with depth of field; the view port where the surgeon views the image; foot pedals to control electrocautery, camera focus and instrument/camera arm clutches, and master control grips that drive the servant robotic arms at the patient's side.

- A *movable cart*, where two instrument arms and the camera arm are mounted. The instruments are cable driven and provide 7 degrees of freedom. This system displays its 3-dimensional image above the hands of the surgeon so that it gives the surgeon the illusion that the tips of the instruments are an extension of the control grips, thus giving the impression of being at the surgical site.

The Zeus system is composed of a surgeon control console and 3 table-mounted robotic arms (Fig. 1.2). The right and left robotic arms replicate the arms of the surgeon, and the third arm is an AESOP (Automated Endoscopic System for Optimal Positioning) voice-controlled robotic endoscope for visualization. In the Zeus system, the surgeon is seated comfortably upright with the video monitor and instrument handles positioned ergonomically to maximize dexterity and allow complete visualization of the operating room environment. The system uses both straight shafted endoscopic instruments similar to conventional endoscopic instruments and jointed instruments with articulating end-effectors and 7 degrees of freedom. Zeus differs from the Da Vinci system in that the AESOP part of ZEUS responds to voice commands. For example, a surgeon might say: "AESOP move right." and the positioning arm then would move right until the "stop" command was given.

The advantages of these systems are many because they overcome many of the obstacles of minimally-invasive surgery. They increase dexterity, restore proper hand-eye coordination and an ergonomic position, and improve visualization. In addition, these systems make surgeries that were technically difficult or unfeasible previously, now possible.

These robotic systems enhance dexterity in several ways. Instruments with increased degrees of freedom greatly enhance the surgeon's ability to manipulate instruments and thus the tissues. These systems are designed so that the surgeon's tremor can be compensated on the end-effector motion through appropriate hardware and software filters. In addition, these systems can scale movements so that large movements of the control grips can be transformed into micromotions inside the patient.

Another important advantage is the restoration of proper hand-eye coordination and an ergonomic position. These robotic systems eliminate the fulcrum effect, making instrument manipulation more intuitive. With the surgeon sitting at a remote, ergonomically

designed workstation, current systems also eliminate the need to twist and turn in awkward positions to move the instruments and visualize the monitor.

By most accounts, the enhanced vision afforded by these systems is remarkable. The 3-dimensional view with depth perception is a marked improvement over the conventional laparoscopic camera views. Also to one's advantage is the surgeon's ability to directly control a stable visual field with increased magnification and maneuverability. All of this creates images with increased resolution that, combined with the increased degrees of freedom and enhanced dexterity, greatly enhances the surgeon's ability to identify and dissect anatomic structures.

However, current surgical robots are not the final answer to surgeon's accuracy demands. In fact, usually surgical robotic systems are teleoperated by the surgeon, as is the case for the Da Vinci surgical system, without any embedded autonomy and therefore performance-bound by the perception and dexterity of the human operator. Moreover, other main drawbacks of teleoperated surgical robots are the significant amount of required training efforts and the necessity for the surgeon to manually perform all the surgical tasks necessary for the operation. This can increase the mental workload and the fatigue for executing the operation and, consequently, decrease the concentration in some, possibly crucial, parts of the operation. In order to overcome this drawback, the possibility of automatically executing some surgical tasks by a robotic system is increasingly being investigated, automating in this way a few phases of the operation [8].

Although it is well known that automation has been successfully used to enhance a great variety of human activities, from aircraft control to manufacturing, the whole area of autonomous interaction of surgical tools with biological tissues is rather unexplored. In the applications where it has been used, automation has increased safety, accuracy, reproducibility, and has decreased human fatigue. Thus, it can be hypothesized that similar benefits could also be gained by introducing automation to specific aspects of surgery, provided that the challenges of this concept are successfully solved. In fact, reports in the general press [9] and in the FDA listing of safety alerts for medical devices indicate the increasing occurrence of potentially dangerous situations during robot-assisted procedures.

The possibility to carry out simple surgical actions automatically has been the subject of academic research. Automatic surgical stapling devices [10] have been developed, even if that devices have no sensing or autonomy. Automatic surgical stapling devices [10] have been developed, even if that devices have no sensing or autonomy. A remotely-controlled catheter guiding robot was used in [11] to automatically perform cardiac ablation. However, an experienced operator is required to perform all the procedures. Several works

(see e.g. [12]) have been done towards the automation of knot tying in suturing tasks, but requiring manual help in several preparatory stages (e.g. grasping the needle). Automatic scissors were proposed in [13], namely the possibility for a surgeon to invoke a third robotic arm to come and automatically cut the thread that he/she is holding. Though all these works constitute successful automation of simple surgical actions, validated and commercially distributed autonomous surgical robots are quite still hard to find. A notable exception is represented by the already cited ROBODOC, a system capable of interventions on rigid tissues in orthopaedic surgery. However, interaction with soft tissues is much more challenging for an autonomous robot, since in this case pre-operative models of the patient might not be available or valid at intra-operative time.

The results of this thesis have been obtained during the FP7 European funded project Intelligent Surgical Robotics (I-SUR), addressing the autonomous execution of basic surgical actions. Such technology will allow in the future the surgeons to focus only on the most difficult aspects of the intervention leaving the basic tasks to the autonomous system.

1.1 The I-SUR project

The main goal of the I-SUR project is to demonstrate that autonomous robotic surgical systems can carry out simple surgical tasks (i.e. needle insertion and wound suturing) effectively and without major intervention by surgeons. To fulfill this goal, innovative solutions (both in terms of technologies and algorithms) have been developed for the following aspects: manufacturing of soft organ models starting from CT images, surgical planning and execution of movements of robot arms in contact with a deformable environment, designing surgical interface minimizing the cognitive load of the surgeon supervising the actions, intra-operative sensing and reasoning to detect normal transitions and unexpected events. All these technologies have been integrated using a component-based software architecture and validated using a surgical robot prototype specifically designed for the autonomous execution of the surgical actions under study, but adaptable to other different surgical tasks.

For an overview of the I-SUR system and preliminary results on the automatic execution of needle insertion for the cryoablation of kidney tumors see [14] and [15].

Consortium The I-SUR team is composed of the following partners, with different tasks and responsibilities:

- ***Fondazione Centro San Raffaele***

Together with the hospital in Verona, is responsible for the surgical action definitions, providing medical competence of all kind throughout the project. It is also responsible for validation of the developed methods on surgical phantoms.

- ***Tallinn University of Technology***

  Develop phantom models of the patient organs with biomechanical properties consistent with the real organs.

- ***Oslo University Hospital***

  Research on sensor configuration and processing for real time monitoring of the surgical action to develop a real time reasoning engine that can make predictions on the action outcome.

- ***ETH Zurich***

  Design and evaluation of a dexterous robotic manipulator equipped with dedicated sensing allowing both autonomous closed-loop control of the device and shared control with feedback to the operator.

- ***University of Verona, University of Modena and Reggio Emilia, University of Ferrara***

  Research related to the control of the surgical platform and the autonomous execution of the surgical actions.

- ***Yeditepe University***

  Develop a communication framework in which the autonomous instrument can convey information to and receive commands from its human supervisor with a multimodal user interface.

1.1.1 Description of the I-SUR surgical tasks

The selection of the procedures to focus on has been made considering the need to provide innovative technologies and methods that could lead to a real improvement in safety, efficacy and accuracy of the surgical interventions by addressing the needs of the surgeons.

The puncture of a target objective inside the human body (e.g. biopsy, tumor ablation, etc.) is described as a very demanding task because of the current pre-operative planning methodology to provide clear indications on the trajectory to be followed from the skin surface to the target organ. Moreover the automation of these surgical tasks involves the solution of challenging problems concerning the required imaging, sensing and control

system to be applied in order to guarantee the autonomy of the robotic system. Simple sutures contribute to increase the operating time for the completion of the intervention and are not exempt from errors and adverse events. High accuracy is required during the execution of this surgical action for guaranteeing patient safety.

Adverse events will be managed by allowing the switching to the teleoperated modality for the most problematic situations.

Puncturing

Puncturing is defined as the act of penetrating a biological tissue with a needle aiming at reaching a specified target point, e.g. to perform a biopsy or perform ablation techniques. To identify a specific procedure for the puncturing task, the focus will be on the insertion of a needle for the cryoablation of kidney tumors.

Percutaneous cryoablation [16] of a small tumoral mass ($\leq 4\ cm$) in the kidney is the least invasive treatment for kidney cancer based on thermal ablation, which aims at destroying neoplastic tissues through a thermal shock caused by short cycles of freezing and thawing.

The percutaneous approach, in contrast with open or laparoscopic surgery, is less invasive, decreases the morbidity and ensures high efficacy by accurately targeting the cancer while preserving the healthy adjacent structures [17], [18].

Percutaneous cryoablation is performed by using a tool called *cryoprobe* which is directly introduced through the skin towards the kidney tumor with the aid of clinical imaging devices (CT, MRI, US) (Fig. 1.3). Cryoprobes are hollow needles, similar to biopsy tools, whose temperature is conditioned by fluids circulating inside, generally Argon for cooling and Helium for heating. The very low temperature generated around the tip allows to freeze the surrounding tissue creating an *iceball* around the tumor. Rapid freezing and thawing cycles induce irreversible damage of the tissue within the iceball.

The most important issue for a puncturing task is to safely and correctly reach the target point. Indeed the precision to insert the needle tip near the center of the tumor is strictly correlated to the success of the treatment. For this reason, the cryoablation procedure can be improved by exploiting the intrinsic high accuracy and repeatability of robotic devices and the pre- and intra-operative images of the patient anatomy.

Suturing

Suturing is the act of closing an opening in a biological tissue by means of a thread. Its purpose is to approximate tissue edges for a functional result, supporting and strengthening wounds until healing increases their tensile strength while minimizing the risks of

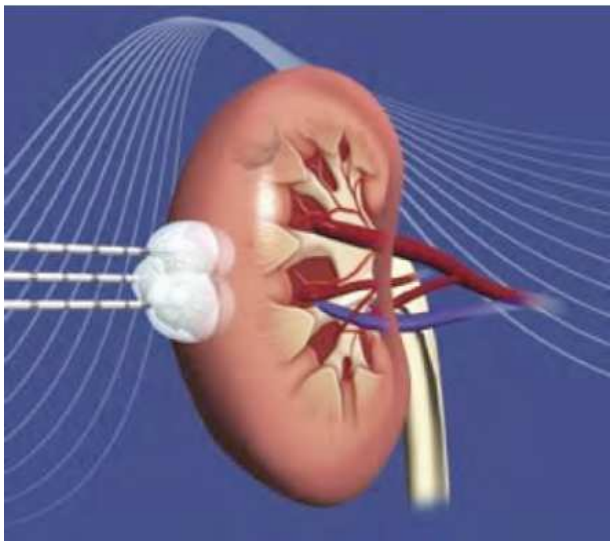


Figure 1.3: Cryoablation is performed with the introduction of hollow hypodermic needles (cryoprobes) through the skin towards the tumor. Each cryoprobe generates around the tip an iceball which is associated to a specific isotherm.

bleeding and infection. The technique may vary depending on tissue's properties, functional result, surgeon's dexterity and preferences.

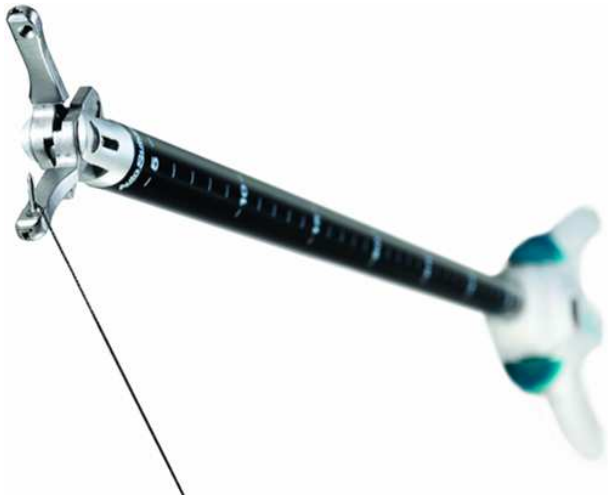
There are many different types of sutures: differences may vary according to tissue and wound type, geometry of the anatomical site and many other aspects. As the simplest use case to analyze the introduction of automation principles in such procedure, in the following will be treated the skin tissue and continuous linear wound. The chosen case study is the suture of a regular opening described as a linear cut on a planar surface. The surface is constituted by a single layer made by uniform tissue and the scope is to join the two edges of the cut.

The reference technique is the *continuous suture* (or simple running suture) which is performed through a series of simple uninterrupted sutures placed in succession without tying or cutting the suture material after each pass. It is one of the most common techniques for several surgery fields (vascular, abdominal, orthopedics...) and can be used on different tissues such as soft organs, subcutaneous layers, vessels, muscles and so on. As mentioned before, the basic technique may vary depending on tissue to be sutured and surgeon's habits.

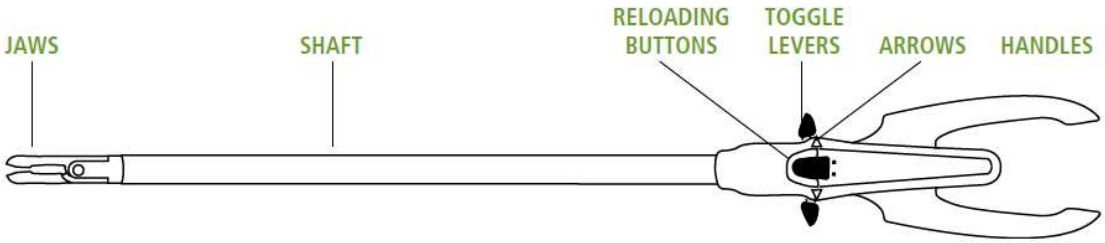
The suturing task will be performed by using a specific semi-autonomous suturing device, the Covidien Endo Stitch system (Fig. 1.4(a)). The Covidien Endo Stitch is a commercial suturing instrument designed to make suturing and the tying of knots easy. The Endo Stitch suturing device, with its integral needle and suturing thread, make it possible to undertake most suturing procedures without the necessity of having to struggle

with loose needles and suture.

The needle is passed back and forth between the instrument's jaws (see 1.4(b) for the nomenclature) by completely squeezing the handles and activating either toggle lever forward or backward. The handles are fully squeezed to close the jaws and are released to open the jaws.



(a) Covidien Endo Stitch



(b) Schematic view and nomenclature

Figure 1.4: The Covidien Endo Stitch system.

The procedure to execute a suture with the Endo Stitch is the following. See Fig. 1.5 for a test of manual suturing with the Endo Stitch.

1. Plan the next stitching point.
2. Place the jaws of the instrument around the first edge to be sutured.
3. Completely squeeze the handles to close on the tissue.
4. Once the jaws are fully closed, activate either toggle lever completely forward or backward (depending on their original position) and the needle is passed from one jaw to the other jaw. Then release the handles.

5. Pull the thread through the tissue.
6. Place the jaws of the instrument around the second edge to be sutured.
7. Bite and switch the needle.
8. Pull the thread.
9. Bite and switch or rotate 180 degrees.
10. Restart from point 1.

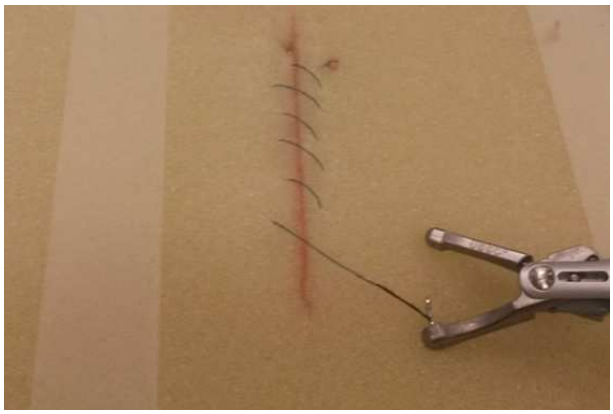


Figure 1.5: A test of a manual suture made with the Covidien Endo Stitch

1.1.2 Surgical tasks automation

In the following, a particular application of the needle insertion task, namely the percutaneous cryoablation of small tumors, will be addressed more intensively, in order to emphasize the potential benefits of the proposed technology and the pre- and intra-operative analysis that it allows. At the same time, that surgical task should be considered as one possible case study, since generality of the presented control algorithms and modularity and reconfigurability of the proposed software architecture allow to easily update the system, provided that the mechanical setup is updated accordingly, for the execution of other surgical tasks.

To emphasize this characteristic, the same methodology and design approach have been applied to automated suture of planar wounds. In this thesis, details about these adaptations will be provided.

Puncturing

As described above, percutaneous cryoablation requires the use of pre- and intra-operative images, acquired by CT, MRI, or US, to insert, through the skin, one or more cryoprobe needles into the tumoral mass to be destroyed and to check the real-time position of the needle tips inside the patient. Thanks to real-time image registration and accurately calibrated mechanical arms, needle insertion can be precisely executed by the robotic system.

Once that needles are correctly inserted, the cryoablation operation involves cycles of freezing and thaw to create an *iceball* covering and killing the tumoral cells, while preserving the healthy tissue and the surrounding abdominal structures.

Major complications refer to bleeding and organ damages caused by the extraction of the probes when the ice ball is not melted enough to release the needles. The evaluation of the force applied to extract a needle from the patient is the only way to detect the melting status of the iceball. Even in this case, robotic assistants equipped with force sensors and intra-operative processing of US images would increase safety margins during completion of the surgical procedure.

To evaluate practical issues and benefits of cryoablation execution by means of automated robots, an experimental setup including the following parts has been prepared and used for experimental validation.

- A UR5 industrial robot [19], holding a US probe thanks to an *ad hoc* adapter,
- A novel robot, specifically developed within the I-SUR project for surgical applications and based on a macro-micro mechanical design approach [20], that will be called *I-SUR robot*. The macro unit is a parallel robot for the coarse positioning, whereas the micro unit is devoted to fine movements and, in this particular task, to hold a cryoablation probe (Fig. 1.6). The surgical tool is mounted on a 6-DOF force/torque sensor needed for control and monitoring purposes.
- A phantom accurately reproducing the human abdomen. More details about the anatomical characteristics of the phantom and its CT and US compatibility are described in [21].
- A PHANTOM Omni [22] haptic device, allowing bilateral teleoperation of the I-SUR robot.

The setup allows to emulate a cryoablation operation, apart from the actual freezing/defreezing cycles, since cryoablation devices could not be installed in the academic

Figure 1.6: Cryoablation straight and 90 degrees probes.

laboratory hosting the experimental setup. A 3-D optical tracking system is used to estimate relative coordinate transformations among the robots and the phantom. Finally, the setup includes an ultrasound imaging device whose images are visualized on a dedicated graphical interface for the surgeons and processed in real-time to detect the position of needle tip, as shown in [23], and provide intra-operative adaptation of robot motion trajectories.

Suturing

To address the suturing task, the following experimental setup has been prepared and used for experimental validation.

- The I-SUR robot as the puncturing task. However, an additional micro unit has been mounted on the macro structure. One arm holds the Covidien Endo Stitch system and is devoted to perform the sequence of stitches, while the other arm holds a tool to remove the suturing thread after each stitch from the workspace of the Endo Stitch.
- A suturing pad with similar mechanical properties as human abdomen cutaneous and subcutaneous layers.
- A couple of PHANTOM Omni haptic devices, allowing bilateral teleoperation of the two end-effector of the robot.
- A PointGrey Bumblebee2 [24] stereo-camera whose images are exploited to recognize the wound edges and the tool tip.

1.2 Contribution and thesis outline

This thesis focuses on the development of control strategies that allow a safe interaction between a robot and the environment in critical tasks, such as the autonomous execution

of surgical actions in robotic surgery. A flexible teleoperation strategy that ensures a stable switch between autonomous and teleoperated mode has been developed to allow the surgeon to undertake the control of the robot in case of unexpected events occur.

A formal design framework is exploited to precisely specify the surgical procedure and translate it into an engineering design. The formal description enables automatic software design of the robotic control system and provides validation-oriented requirements that must be addressed during functional tests. For validation and proof of concepts, the proposed automation system has been implemented into a component-based distributed software architecture.

In particular, the main contributions of this thesis are:

- A novel tank-based time-varying admittance controller allowing to adapt the interactive behavior of the robot while preserving passivity. The results are then extended to implement a variable impedance controller.

- A novel flexible and passivity-based teleoperation architecture that ensure a stable switch between autonomous and teleoperated modes and safely compensates for the kinematic mismatch between the master and the slave (i.e. the surgical robot).

- A novel teleoperation strategy for teleoperation of a dual arms manipulator. The strategy allows to change the direction of the assistive force generated by guidance virtual fixtures while preserving the passivity of the overall system. The direction of the assistive force is changed in order to maximize the efficiency of the teleoperation system.

- The integration of sensing, cognition and control capabilities into a modular software system, whose architectural properties allow to enhance reconfigurability and reusability of its main components. Particular care is given to the implementation of robot motion planning, control and supervision, which is the most critical part of the system.

- An experimental validation of the control strategies and of the software architecture using the I-SUR robotic surgery platform and realistic phantom models. The case studies taken into consideration are percutaneous kidney cryoablation and suturing of a planar wound. on a kidney crioablation scenario

The thesis is organized as follows:

Chapter 2 describes the passivity-based interactive control strategies that allow to implement safe and stable time-varying interactive behaviors and a transient-free kinematically compensated bilateral teleoperation of a surgical robot. The results presented

in [25] are extended to develop admittance and impedance control strategies that allow to implement time-varying admittance behaviors in which the controlled inertia, stiffness and damping can be changed in a passive way. Then, energy tanks and the two-layer approach proposed in [26] and [27] are exploited to develop a novel teleoperation architecture for safely connecting master and slave and compensating for kinematic mismatch during the switch from autonomous to teleoperated mode. The results obtained are integrated into a strong framework and presented in [28]. In the final part of the chapter, will be shown how it is always possible to embed virtual fixtures on the master side for assisting the user during teleoperation of a dual arms system without violating the passivity constraint. The results obtained are presented in [29].

Chapter 3 presents the component-based distributed software architecture that has been developed to implement the control strategies described in the previous chapter and to integrate the different parts of the whole surgical system. Following the approach described in [30] and extended in [31], a robot control and coordination framework for the automation of simple surgical tasks is presented. The design specifications are formalized using a requirements engineering approach and the state machines for the control of the robots involved in the operations are derived. The proposed architecture is then implemented using component-based design tools, to properly handle the distributed nature of the system. Further results on the control and simulation framework proposed in this chapter are presented in [32].

Chapter 4 describes the experimental validation of the proposed control strategies and software architecture. First, the robotic platform designed within the I-SUR project is described. Then, implementation of the control methods and software architecture is presented to show how the two particular surgical tasks already presented in this introductive chapter (puncturing and suturing) could be automatized. To address the detailed planning of the puncturing task, a planning tool (i.e. the *Cryo-planner*) has been developed following the results shown in [33] and [34]. Preliminary results on the automatic execution of needle insertion for the cryoablation of kidney tumors are presented in [14] and [15].

Chapter 2

Interaction control for robotic surgery

This chapter describes a novel time-varying admittance controller that allows to adapt the interactive behavior of the robot, while preserving passivity. When autonomously executing a surgical task, a robot has to interact with a mostly unknown and usually time-varying (deformable or non homogeneous) environment. The proposed control strategy ensures a stable and safe interaction between the robot and the unknown environment it has to interact with. Moreover, in the event of unexpected events taking place, the surgeon has to undertake the control of the robot to drive it to a safe configuration. Thus, a novel teleoperation architecture to safely connecting master and slave when switching from autonomous to teleoperated mode will be presented. Finally, in the last section, the generation of assistive forces to implement guidance virtual fixtures for a teleoperated dual arms system will be discussed.

2.1 Introduction

Introducing some form of autonomy in robotic surgery is being considered by the medical community to better exploit the potential of robots in the operating room. The initial steps of the automation process require to provide the robot with the necessary dexterity and a stable and smooth behavior of the surgical tool. Moreover, the human intervention needs always to be considered, because of unforeseen situations that cannot be addressed autonomously. In these situations, the surgeon is expected to take control of the operation by means of a teleoperation scheme.

When autonomously executing a surgical task, a robot faces a very challenging situation. In fact, the environment the robot has to interact with (i.e. the body of the patient) is mostly unknown (only some information can be obtained from pre-operative imaging) and usually time-varying (deformable or non homogeneous). Furthermore, the robot may need to change its interactive behavior online, because the operating conditions change. For example, during the needle insertion task, the robot has to be stiff when

penetrating the skin, while it has to behave more softly during the insertion. The fact that each patient is different and that, therefore, it is often not possible to determine a standard pattern for the desired behavior, increases task complexity. Finally, the safety of the patient is always the primary concern. As shown in [35], a safe reactive behavior of the robot to some selected complications can be planned in advance. Nevertheless, in the event of unexpected events taking place, it is necessary to let the surgeon take over the control of the robot to drive it to a safe configuration.

Impedance and admittance control strategies [36] are highly suitable candidates to ensure a stable and safe interaction with unknown environments. They allow to directly shape the behavior by which the robot interacts with the environment. Unfortunately, standard impedance and admittance control schemes cannot guarantee a stable interaction when time-varying interactive behaviors need to be implemented.

Controlling the robot in teleoperation brings other challenges. Bilateral teleoperation has been proven very effective for allowing a human to drive a remote robot [37] and, therefore, it is also a good candidate for allowing the surgeon to take over the control of the robot. Nevertheless, when the surgeon switches the robot from an autonomous mode to a teleoperated mode, there is likely a kinematic mismatch between the pose of the master console and that of the surgical robot (i.e. the slave robot). This mismatch can impose a high workload on the surgeon to mentally compensate the offset, and can therefore lead to risks for the patient because of unintentional motions transmitted to the robot. Furthermore, by simply switching on standard bilateral controllers (e.g. position-position, position-force [38]) in a mismatched situation, unexpected transients on the slave could be experienced. These problems are highly undesirable because the teleoperation mode is activated during critical situations, and mistakes in the teleoperation can cause severe injury to the patient. While many bilateral teleoperation control strategies ensuring an efficient and safe behavior have been proposed in the literature [38], very few bilateral teleoperation systems capable of safely switching between autonomous and teleoperated modes and compensating kinematic mismatches exist.

Most of the surgical robots, either commercially available or developed within academic research, are teleoperated and have multiple arms. As already introduced, the teleoperation control scheme is actually mandatory during operations on real patients, since human surgeons must always be able to exert a direct command to the robotics arms. A mechanical structure with multiple arms is generally acknowledged as a successful emulation of human features. On the other hand, teleoperating a multi-arm robot is a complex task that surgeons are able to accomplish only after a long training, which is a motivation for the interest in surgical robots automation or, at least, in the devel-

opment of control techniques to assist users of such teleoperated systems. In particular, an approach commonly proposed in the teleoperation literature is the implementation of virtual fixtures [39], generally defined as assistive forces, applied to the haptic device used as a teleoperation master. These virtual fixtures can be designed to restrict the motion of the master to desired subspaces or volumes in its operational space, or to guide the user towards a desired target, that can be either a specific operating point or a more complex geometric path. However, when an attractive fixture is exploited for driving the user towards a certain path, it can happen that the repulsive forces generated by the fixtures devoted to collision avoidance take the operator far from the desired path leading to an inefficient task execution.

In this chapter, the port-Hamiltonian framework [40] and the concept of energy tanks [41], [26] will be exploited for modeling the surgical robotic architecture and for using the (virtual) energy circulating in the controlled system in a flexible and passivity preserving way.

The following control strategies will be presented:

- A passivity-based interactive control architecture that allows to implement safe and stable time-varying admittance behaviors in which the controlled inertia, stiffness and damping can be changed.
- A novel flexible and passivity-based teleoperation architecture that ensures a safe switch between autonomous and teleoperated modes and safely compensates for the kinematic mismatch between the master and the slave (i.e. the surgical robot).
- A novel teleoperation strategy for a dual arms system that allows to change the direction of the assistive force generated by the virtual fixture while preserving the passivity of the overall system.

The proposed control strategies, addressing the mentioned issues related to robotic surgery, have been tested and validated using the surgical robot prototype specifically designed within the I-SUR project. Experimental results will be shown in Chapter 4.

2.1.1 Outline

The outline of the Chapter is as follows. Section 2.2 provides some background on port-Hamiltonian systems and energy tanks. Section 2.3 presents the tank-based variable admittance controller and its adaptation for implementing a variable impedance controller, while in Section 2.4 the strategy to safely switch to teleoperation is presented. Finally, Section 2.5 describes the teleoperation strategy for a dual arms system with exploitation of guidance virtual fixtures.

2.2 Background on port-Hamiltonian systems and energy tanks

This section provides some background on port-Hamiltonian systems and on energy tanks. For a more detailed treatment the reader is referred to [40], [42], [26].

The port-Hamiltonian framework is a generalization of standard Hamiltonian mechanics, where energetic characteristics and power exchange between sub-systems are clearly identified. All physical systems, even multi-domain, can be represented using the port-Hamiltonian formalism. The most common representation of a port-Hamiltonian system is:

$$\begin{cases} \dot{x} = [J(x) - R(x)] \dfrac{\partial H}{\partial x} + g(x)u \\ y = g^T(x) \dfrac{\partial H}{\partial x} \end{cases} \tag{2.1}$$

where $x \in \mathbb{R}^n$ is the state vector and $H(x) : \mathbb{R}^n \to \mathbb{R}$ is the lower bounded Hamiltonian function representing the amount of energy stored in the system. Matrices $J(x) = -J(x)^T$ and $R(x) \geq 0$ represent the internal energetic interconnections and the dissipation of the port-Hamiltonian system, respectively, and $g(x)$ is the input matrix. The input u and the output y are dual variable and their product is (generalized) power. The pair (u, y) is called power port and is the means by which the system can energetically interact with the external world. The product $u^T y$ represents the power exchanged by the system with the external world.

It can be easily shown that the following equality holds [40]:

$$\dot{H}(x) + \frac{\partial^T H}{\partial x} R(x) \frac{\partial H}{\partial x} = u^T(t) y(t) \tag{2.2}$$

This means that the power supplied to the system is either stored or dissipated, namely that a port-Hamiltonian system is passive with respect to the pair (u, y).

Let

$$D(x) = \frac{\partial^T H}{\partial x} R(x) \frac{\partial H}{\partial x} \geq 0 \tag{2.3}$$

indicate the power dissipated by the system. As pointed out in [43], $D(x)$ represents a *passivity margin*: the larger $D(x)$, higher the passivity of the system. In other words, the larger the passivity margin, the more the system can absorb the energy generated by non passive actions (e.g. changing the stiffness in a visco-elastic coupling, as discussed later) while preserving its passivity.

Energy tanks, firstly proposed in [41], exploit this concept for building flexible and passivity preserving controllers. The energy dissipated by the system is stored in a (virtual) energy tank and can be reused for implementing any desired control action in a passivity preserving way.

More formally, the dynamics of a port-Hamiltonian system endowed with a tank is given by:

$$\begin{cases} \dot{x} = [J(x) - R(x)] \dfrac{\partial H}{\partial x} + g(x)u \\ \dot{x}_t = \dfrac{\sigma}{x_t} D(x) + \dfrac{1}{x_t}(\sigma P_{in} - P_{out}) + u_t \\ y_1 = \begin{pmatrix} y \\ y_t \end{pmatrix} \end{cases} \tag{2.4}$$

where $x_t \in \mathbb{R}$ is the state associated with the energy storing tank and

$$T(x_t) = \frac{1}{2} x_t^2 \tag{2.5}$$

is the energy stored in the tank. $P_{in} \geq 0$ and $P_{out} \geq 0$ are incoming and outgoing power flows that the tank can exchange with other tanks. The pair (u_t, y_t) is a power port that the tank can use to exchange energy with the external world and $y_t = \frac{\partial T}{\partial x_t} = x_t$. The parameter $\sigma \in \{0, 1\}$ is used for bounding the amount of energy that can be stored in the tank. The following power balance can be easily derived from (2.4):

$$\dot{T} = \sigma D(x) + \sigma P_{in} - P_{out} + u_t^T y_t \tag{2.6}$$

which means that, if $\sigma = 1$, the tank stores the power dissipated by the system $D(x)$ and the incoming power flow P_{in}, while the outgoing power flow P_{out} is released. Furthermore, energy can be injected in / extracted from the tank via the power port (u_t, y_t). In order to avoid singularities in (2.4), some energy must always be present in the tank (i.e. $x_t \neq 0$). Thus, it is necessary to set an arbitrarily small threshold $\varepsilon > 0$ representing the minimum amount of energy that needs to be always stored. The tank has to be initialized and managed in such a way that $T(x_t(0)) > \varepsilon$ and energy extraction is prevented if $T(x_t) \leq \varepsilon$. Finally, it is necessary to set an upper bound on the amount of energy that can be stored in the tank, in order to avoid the problems described in [44]. In fact, if there is no bound, the energy available can become very big as time increases and, even if the system remains passive, it would be possible to implement behaviors which are unstable in practice. Thus, σ is set using the following policy:

$$\sigma = \begin{cases} 1 & if\ T(x_t) \leq \bar{T} \\ 0 & otherwise \end{cases} \tag{2.7}$$

where $\bar{T} > 0$ is a suitable, application dependent upper bound on the energy that can be stored in the tank.

The energy stored in the tank can be exploited for passively implementing any desired input $w \in \mathbb{R}^n$ to the port-Hamiltonian system the tank is associated to. This can be

done by joining the power ports (u, y) and (u_t, y_t) through the following power preserving interconnection:

$$\begin{cases} u = \dfrac{w}{x_t} y_t = \dfrac{w}{x_t} x_t = w \\ u_t = -\dfrac{w^T}{x_t} y \end{cases} \tag{2.8}$$

implying the following balance

$$u^T y = -u_t y_t \tag{2.9}$$

When using (2.8) the energy supplied to/extracted from the port-Hamiltonian system for implementing the desired input is exactly equal to the energy extracted from / supplied to the tank. This intuitively means that no energy is generated and that the desired input can be implemented in a way to preserve passivity as long as some energy is stored in the tank (i.e. $T(x_t) > \varepsilon$). For a more formal treatment see [42], [26].

2.3 Impedance and admittance control with variable stiffness

Admittance control and impedance control [45] are very effective control schemes for implementing a desired interaction behavior. Loosely speaking, impedance control is more suitable for backdrivable robots while admittance control is more suitable for stiff robots. The robot developed within the I-SUR project has a stiff and not backdrivable structure. Therefore, in Section 2.3.1 will be shown how to exploit tanks for implementing a variable admittance control. However, all the results developed for the admittance control can be easily adapted for implementing a variable impedance control, as will be shown in Section 2.3.2.

2.3.1 Variable admittance control

A standard admittance control scheme is reported in Fig. 2.1. Given a desired interaction model, namely a dynamic relation between the applied force and the pose error, given the external force and the desired pose setpoint, the corresponding position of the robot is generated and tracked by the robot by means of a lower level motion controller. The motion controller has to be designed and tuned to minimize the tracking error and optimize the dynamic response, so that the actual pose $x \in \mathbb{R}^n$, $n \leq 6$, of the robot end-effector can be assumed identical to its reference position $x_{ref} \in \mathbb{R}^n$. Thus, in the following will be assumed $x = x_{ref}$.

A very common and useful example of admittance control is a nonlinear feedback law that makes a manipulator equivalent to a multidimensional mass-spring-damper system

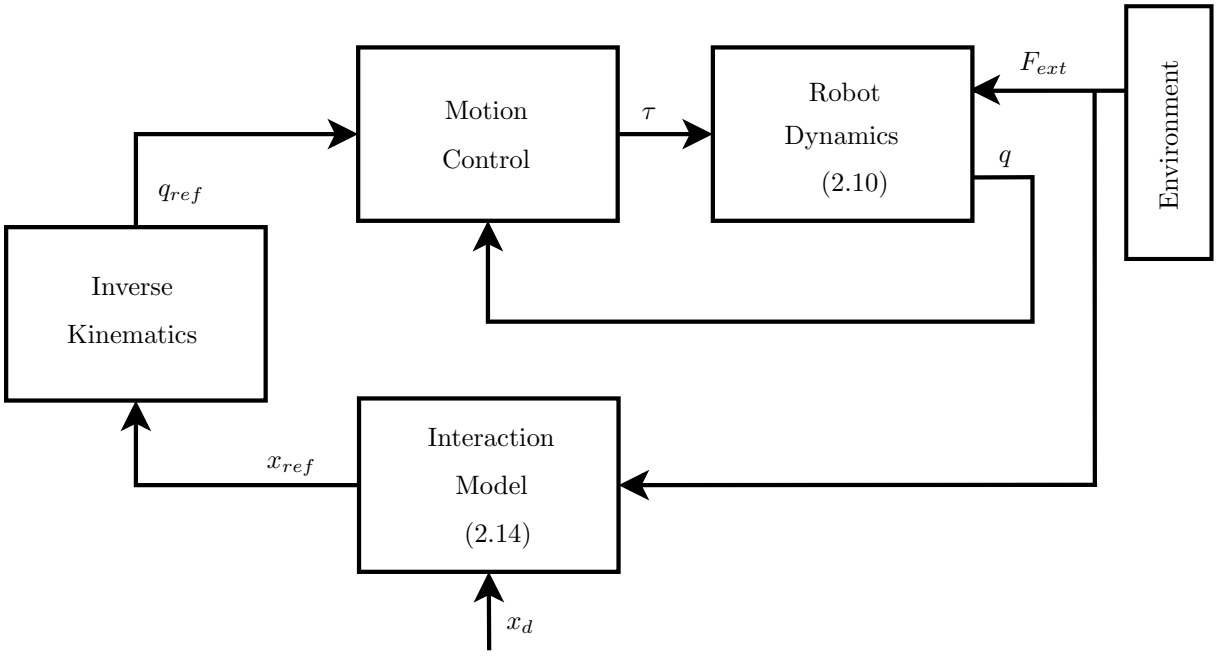


Figure 2.1: Control scheme for admittance control with underlying motion controller. The solution of the interaction model (2.14) with the inputs F_{ext} and x_d provides the value x_{ref} which the position-controlled robot must follow.

with desired inertia, stiffness and damping properties.

More formally, consider the following Euler-Lagrange dynamic model of a m-DOFs, $m \geq n$, fully actuated manipulator with revolute joints:

$$M(q)\ddot{q} + C\left(q, \dot{q}\right)\dot{q} + g(q) = \tau + J(q)^T F_{ext} \tag{2.10}$$

where $q \in \mathbb{R}^m$ is the vector of joint variables, $M(q) \in \mathbb{R}^{m \times m}$ is the positive definite inertia matrix, $C(q, \dot{q})\dot{q} \in \mathbb{R}^m$ is the vector of Coriolis and centrifugal torques and $g(q) \in \mathbb{R}^m$ denotes the gravity term. The vector $\tau \in \mathbb{R}^m$ represents the controlled joint torques and the term $J(q)^T F_{ext} \in \mathbb{R}^m$ represents the torque applied to the joints because of the external wrench $F_{ext} \in \mathbb{R}^n$, applied to the end-effector. $J(q) \in \mathbb{R}^{n \times m}$ indicates the Jacobian of the manipulator.

If the interaction between the end-effector and the environment needs to be controlled, it is useful to build the model of the robot in the task space:

$$\Lambda(x)\ddot{x} + \mu\left(x, \dot{x}\right)\dot{x} + F_g(x) = F_\tau + F_{ext} \tag{2.11}$$

where $x = f(q)$ is the pose of the end-effector, obtained from the joint positions $q \in \mathbb{R}^m$ through the forward kinematic map $f(\cdot)$. The matrices $\Lambda(x) = \Lambda^T(x) > 0$ and $\mu(x, \dot{x}) \in \mathbb{R}^{n \times n}$ are the $n-$dimensional positive definite inertia matrix and the centrifugal and Coriolis

terms, given by

$$\Lambda(x) = (J(q)M(q)^{-1}J(q)^T)^{-1} \tag{2.12}$$

$$\mu(x,\dot{x}) = J(q)^{-T}[C(q,\dot{q}) - M(q)J(q)^{-1}\dot{J}(q)]J(q)^{-1} \tag{2.13}$$

For the properties of Euler-Lagrange models, the matrix $\dot{\Lambda}(x) - 2\mu(x,\dot{x})$ is skew-symmetric.

The control wrench F_τ is set by the motion controller for implementing a desired interactive behavior.

Remark 1. *In the case of a kinematically redundant manipulator, it is assumed that the redundancy is addressed in the low-level control. Thus, the implementation of the high-level control in the task space as described in the following would not be affected by redundancy.*

Let $x_d(t) \in \mathbb{R}^m$ be a desired configuration for the end-effector. The goal of the admittance controller is to regulate the robot in such a way the relationship between the pose error $\tilde{x}(t) = x(t) - x_d(t)$ and F_{ext} is given by

$$\Lambda_d\ddot{\tilde{x}} + D_d\dot{\tilde{x}} + K_d\tilde{x} = F_{ext} \tag{2.14}$$

where Λ_d, D_d and K_d are the n-dimensional symmetric and positive definite inertia, damping and stiffness matrices characterizing the interactive behavior.

The controlled robot behaves as (2.14) and it is passive with respect to the pair $(F_{ext}, \tilde{x})$. In fact, consider

$$V(\tilde{x},\dot{\tilde{x}}) = \frac{1}{2}\dot{\tilde{x}}^T\Lambda_d\dot{\tilde{x}} + \frac{1}{2}\tilde{x}^T K_d\tilde{x} \tag{2.15}$$

as a non negative storage function. It follows that:

$$\dot{V} = \dot{\tilde{x}}^T\Lambda_d\ddot{\tilde{x}} + \tilde{x}^T K_d\dot{\tilde{x}} \tag{2.16}$$

Using (2.14) in (2.16) one can obtain:

$$\dot{V} = \dot{\tilde{x}}^T F_{ext} - \dot{\tilde{x}}^T D_d\dot{\tilde{x}} \leq \dot{\tilde{x}}^T F_{ext} \tag{2.17}$$

which implies the following passivity condition

$$V(t) - V(0) \leq \int_0^t \dot{\tilde{x}}^T(\tau)F_{ext}(\tau)d\tau \tag{2.18}$$

The passivity of the controlled behavior is a crucial characteristic of admittance control. In fact, passivity is a sufficient condition for ensuring a stability of the controlled

robot both in free motion and during the interaction with any passive, possibly unknown, environment [40]. Furthermore, in case of free motion (i.e. $F_{ext} = 0$), using (2.15) as a Lyapunov function and considering (2.17), it can be shown by a straightforward application of LaSalle's invariance principle that $\tilde{x}(t) \mapsto 0$, namely that the robot asymptotically tracks the desired reference.

In order to reproduce the behavior of the surgeon during operations [46], it may be useful to change over time the parameters of the admittance control. In this way, the admittance control becomes as unconstrained as possible and provides high flexibility to the interactive robot.

Let $\Lambda_d(t)$, $D_d(t)$ and $K_d(t)$ be the time-varying inertia, damping and stiffness matrices. The desired interaction model is given by:

$$\Lambda_d(t)\ddot{\tilde{x}} + D_d(t)\dot{\tilde{x}} + K_d(t)\tilde{x} = F_{ext} \tag{2.19}$$

In order to preserve their physical meaning, $\Lambda_d(t)$, $D_d(t)$ and $K_d(t)$ are assumed symmetric and positive definite for all $t \geq 0$.

The main drawback due to the introduction of a variable interaction model in an admittance control scheme is the loss of passivity of the controlled robot. In fact, consider the total energy of the controlled system as a natural non negative storage function:

$$\mathcal{V}(\tilde{x}, \dot{\tilde{x}}) = \frac{1}{2}\dot{\tilde{x}}^T \Lambda_d(t)\dot{\tilde{x}} + \frac{1}{2}\tilde{x}^T K_d(t)\tilde{x} \tag{2.20}$$

It follows that:

$$\dot{\mathcal{V}} = \dot{\tilde{x}}^T \Lambda_d(t)\ddot{\tilde{x}} + \frac{1}{2}\dot{\tilde{x}}^T \dot{\Lambda}_d(t)\dot{\tilde{x}} + \tilde{x}^T K_d(t)\dot{\tilde{x}} + \frac{1}{2}\tilde{x}^T \dot{K}_d(t)\tilde{x} \tag{2.21}$$

Computing $\ddot{\tilde{x}}$ from (2.19) and replacing it in (2.21) one can obtain:

$$\dot{\mathcal{V}} = \dot{\tilde{x}}^T F_{ext} + \left[\frac{1}{2}\dot{\tilde{x}}^T(\dot{\Lambda}_d(t) - 2D_d(t))\dot{\tilde{x}} + \frac{1}{2}\tilde{x}^T \dot{K}_d(t)\tilde{x}\right] \tag{2.22}$$

Because of the variability of the stiffness and of the inertia, the term between the brackets can be positive and the system can produce energy and, therefore, the balance in (2.18) is not always satisfied. Consequently, the passivity of the interaction model does not hold anymore and a stable interaction and an asymptotic tracking in free motion are no longer guaranteed. In the rest of the section will be shown how it is possible, using energy tanks, to implement the flexible interaction model (2.19) while preserving the passivity of the controlled robot.

The main idea for implementing a variable admittance controller is to endow the desired interaction model with a tank for storing the energy dissipated by the system and for re-using it for implementing passivity threatening variable behaviors in a passive way.

The variable inertia and stiffness matrices are split into the sum of a constant term and of a variable term:

$$\begin{cases} \Lambda_d(t) = \Lambda_c + \Lambda_v(t) \\ K_d(t) = K_c + K_v(t) \end{cases} \tag{2.23}$$

where Λ_c, K_c, $\Lambda_v(t)$ and $K_v(t)$ are symmetric positive definite matrices. The constant terms may represent the minimum implementable inertia and stiffness but, in general, it is sufficient that they are constant, symmetric and positive definite and they can represent other significant values.

Considering (2.23), it is possible to reformulate the interaction model (2.19) as a port-Hamiltonian system:

$$\begin{cases} \begin{pmatrix} \dot{\tilde{x}} \\ \dot{\tilde{p}} \end{pmatrix} = \begin{pmatrix} 0 & I \\ -I & -K_d(t) \end{pmatrix} \begin{pmatrix} \frac{\partial H_c}{\partial \tilde{x}} \\ \frac{\partial H_c}{\partial \tilde{p}} \end{pmatrix} + \begin{pmatrix} 0 \\ I \end{pmatrix} F_{ext} + \begin{pmatrix} 0 \\ I \end{pmatrix} (-K_v(t)\tilde{x} - \Lambda_v(t)\ddot{\tilde{x}}) \\ y = \dot{\tilde{x}} \end{cases} \tag{2.24}$$

where $\tilde{p} = \Lambda_c \dot{\tilde{x}}$ and

$$H_c(\tilde{x}, \tilde{p}) = \frac{1}{2}\tilde{x}^T K_c \tilde{x} + \frac{1}{2}\tilde{p}^T \Lambda_c^{-1} \tilde{p} \tag{2.25}$$

As is evident in (2.24), the variability of the inertia and of the stiffness matrix produces an extra input in the port-Hamiltonian dynamics and this input can inject energy into the system destroying its passivity. In order to overcome this problem, the port-Hamiltonian interaction model can be augmented with an energy tank. In this manner, the dissipated energy can be tracked and exploited for passively implementing the extra input due to the variability of the admittance parameters. Thus, the following augmented port-Hamiltonian interaction model is proposed:

$$\begin{cases} \begin{pmatrix} \dot{\tilde{x}} \\ \dot{\tilde{p}} \end{pmatrix} = \begin{pmatrix} 0 & I \\ -I & -K_d(t) \end{pmatrix} \begin{pmatrix} \frac{\partial H_c}{\partial \tilde{x}} \\ \frac{\partial H_c}{\partial \tilde{p}} \end{pmatrix} + \begin{pmatrix} 0 \\ I \end{pmatrix} F_{ext} + \begin{pmatrix} 0 \\ I \end{pmatrix} w \\ \dot{x}_t = \frac{\sigma}{x_t} \tilde{p}^T \Lambda_c^{-1} K_d(t) \Lambda_c^{-1} \tilde{p} - \frac{w^T}{x_t} \dot{\tilde{x}} \\ y = \dot{\tilde{x}} \end{cases} \tag{2.26}$$

where $x_t \in \mathbb{R}$ and $T(x_t) = \frac{1}{2}x_t^2$ are the state and the energy function of the tank respectively. The parameter $\sigma \in \{0, 1\}$ is used to disable the dissipated energy storage in case a maximum limit is reached. The tank initial state is set to $x_t(0)$ such that $T(x_t(0)) > \varepsilon$ and

$$w(t) = \begin{cases} (-K_v(t)\tilde{x} - \Lambda_v(t)\ddot{\tilde{x}}) & \text{if } T(x_t) > \varepsilon \\ 0 & \text{otherwise} \end{cases} \tag{2.27}$$

The energy stored in the tank is exploited for implementing the input due to the variability of the stiffness and inertia parameters. If there is some energy stored in the tank, the

desired interaction model is implemented otherwise the variable parts of the admittance parameters are not implemented. This means that the priority is always given to passivity: if the variable interaction model cannot be implemented using the available energy, then a passive constant behavior, associated to the constant parameters Λ_c and K_c, is implemented.

The power extracted from the tank for implementing the desired behavior is given by:

$$u_t y_t = \frac{w^T}{x_t}\dot{\tilde{x}}x_t = -\tilde{x}^T K_v(t)\dot{\tilde{x}} - \ddot{\tilde{x}}^T \Lambda_v(t)\dot{\tilde{x}} \tag{2.28}$$

Thus, if the variability of the inertia and of the stiffness with respect to Λ_c and K_c is small and/or if the tracking error is small, the energy necessary for implementing the desired behavior is small and, therefore, it is very likely that the energy stored in the tank is sufficient for passively reproducing the desired behavior. Furthermore, it is possible to temporarily increase the value of the desired damping for storing more energy into the tank and increasing the flexibility of the system at the price of a (temporarily) over-damped behavior.

Remark 2. *In order to shape both the inertia and the stiffness matrices, an acceleration measurement (or estimation) is necessary. If this is not available, the desired inertia should be constant but it is possible to shape the stiffness matrix using only position measurements.*

Proposition 1. *The augmented port-Hamiltonian interaction model* (2.26) *is passive with respect to the pair* $(F_{ext}, \dot{\tilde{x}})$.

Proof. Consider as a storage function the total energy of the system:

$$W(t) = H_c(\tilde{x}, \tilde{p}) + T(x_t) = \frac{1}{2}\tilde{x}^T K_c \tilde{x} + \frac{1}{2}\tilde{p}^T \Lambda_c^{-1}\tilde{p} + \frac{1}{2}x_t^2 \tag{2.29}$$

The interaction model in (2.26) can be rewritten as:

$$\begin{cases} \begin{pmatrix} \dot{\tilde{x}} \\ \dot{\tilde{p}} \\ \dot{x}_t \end{pmatrix} = \left[\begin{pmatrix} 0 & I & 0 \\ -I & 0 & \frac{w}{x_t} \\ -\frac{w^T}{x_t} & 0 & 0 \end{pmatrix} - \begin{pmatrix} 0 & 0 & 0 \\ 0 & K_d(t) & 0 \\ -P(t) & 0 & 0 \end{pmatrix} \right] \begin{pmatrix} \frac{\partial W}{\partial \tilde{x}} \\ \frac{\partial W}{\partial \tilde{p}} \\ \frac{\partial W}{\partial x_t} \end{pmatrix} + \begin{pmatrix} 0 \\ I \\ 0 \end{pmatrix} F_{ext} \\ y = \dot{\tilde{x}} \end{cases} \tag{2.30}$$

where $P(t) = \frac{\sigma}{x_t}\tilde{p}^T \Lambda_c^{-1} K_d(t)$. Considering (2.30), from simple computations it follows that:

$$\dot{W} = -\tilde{p}^T \Lambda_c^{-1} K_d(t) \Lambda_c^{-1}\tilde{p} + \sigma \tilde{p}^T \Lambda_c^{-1} K_d(t) \Lambda_c^{-1}\tilde{p} + F_{ext}^T \dot{\tilde{x}} = \tag{2.31}$$

$$= -\tilde{p}^T \Lambda_c^{-1}(K_d(t) - \sigma K_d(t))\Lambda_c^{-1}\tilde{p} + F_{ext}^T \dot{\tilde{x}} \tag{2.32}$$

Since $\sigma \in \{0, 1\}$ and since $K_d(t) > 0$, then $(K_d(t) - \sigma K_d(t)) \geq 0$ and, therefore, $\dot{W} \leq F_{ext}^T \dot{\tilde{x}}$ which implies:

$$W(t) - W(0) \leq \int_0^t F_{ext}^T \dot{\tilde{x}} d\tau \tag{2.33}$$

which proves passivity. □

Thanks to the energy tank, it is possible to implement any variable interaction model satisfying the passivity constraint that guarantees a stable behavior and asymptotic tracking in free motion.

2.3.2 Variable impedance control

The results obtained in Section 2.3.1 can be easily adapted for implementing a variable impedance control. Indeed, the main difference between admittance and impedance control is in the causality of the interaction model and, therefore, in the control output (i.e. position/velocity for admittance controllers, force for impedance controllers). A standard impedance control scheme is shown in Fig. (2.2).

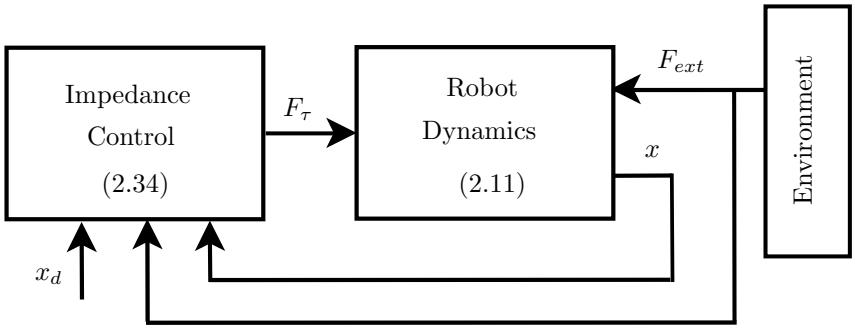


Figure 2.2: Control scheme for impedance control. The impedance control (2.34) with the inputs F_{ext}, x and x_d provides the forces F_τ that the robot must apply.

As for the admittance control, let $x_d(t) \in \mathbb{R}^m$ be a desired configuration for the end-effector. The goal of the impedance control is again to regulate the robot in such a way the relationship between $\tilde{x}(t) = x(t) - x_d(t)$ and F_{ext} is given by (2.14), where Λ_d, D_d and K_d are, respectively, the desired inertia, damping and stiffness matrices.

It can be easily seen that (2.14) can be achieved by setting in (2.11):

$$F_\tau = F_g(x) + \Lambda(x)\ddot{x}_d + \mu(x, \dot{x})\dot{x} - \Lambda(x)\Lambda_d^{-1}(D_d\dot{\tilde{x}} + K_d\tilde{x}) + (\Lambda(x)\Lambda_d^{-1} - I)F_{ext} \tag{2.34}$$

In order to achieve (2.14), it is necessary to feedback the external force F_{ext}. In some application scenarios, like some procedures of robotic surgery, it is not always possible to

put a force sensor on the end-effector (e.g. for sterility reasons). As proposed in [47], the direct feedback of F_{ext} can be avoided setting $\Lambda_d = \Lambda(x)$ and

$$F_\tau = F_g(x) + \Lambda(x)\ddot{x}_d + \mu\left(x, \dot{x}\right)\dot{x}_d - D_d\dot{\tilde{x}} - K_d\tilde{x} \tag{2.35}$$

Using (2.35) in (2.11) the following mechanical impedance is obtained:

$$\Lambda(x)\ddot{\tilde{x}} + (D_d + \mu(x, \dot{x}))\dot{\tilde{x}} + K_d\tilde{x} = F_{ext} \tag{2.36}$$

This kind of impedance is suitable for applications where the velocity is low (e.g. surgical robots executing autonomously interactive tasks on patients) and, consequently, the effect of the term $\mu(x, \dot{x})$ is negligible.

The controlled system reported in (2.36) is passive with respect to the pair $(F_{ext}, \dot{\tilde{x}}(t))$ using the storage function

$$V(\tilde{x}, \dot{\tilde{x}}) = \frac{1}{2}\dot{\tilde{x}}^T\Lambda(x)\dot{\tilde{x}} + \frac{1}{2}\tilde{x}^T K_d\tilde{x} \tag{2.37}$$

In fact, it follows that

$$\dot{V} = \dot{\tilde{x}}^T\Lambda(x)\ddot{\tilde{x}} + \frac{1}{2}\dot{\tilde{x}}^T\dot{\Lambda}(x)\dot{\tilde{x}} + \tilde{x}^T K_d\dot{\tilde{x}} \tag{2.38}$$

Computing $\ddot{\tilde{x}}$ from (2.36), replacing it in (2.38) and considering that $\dot{\Lambda}(x) - 2\mu(x, \dot{x})$ is skew symmetric one obtains again (2.17) which implies the passivity condition (2.18). Since (2.14) can be obtained from (2.36) by setting $\Lambda(x) = \Lambda_d$ and $\mu(x, \dot{x}) = 0$, it follows that (2.14) is also passive with respect to the storage function (2.37) and the pair $(F_{ext}, \dot{\tilde{x}}(t))$.

The goal is still to implement impedance models characterized by a variable stiffness as, for example, the one of a surgeon during a puncturing task [46]. In the following, will be considered the impedance model reported in (2.36). Since (2.14) is a special case of (2.36), all the results that will be obtained will hold also for the standard linear impedance model. Formally, if $K_d(t)$ is the time-varying impedance matrix, the desired impedance model is given by:

$$\Lambda(x)\ddot{\tilde{x}} + (D_d + \mu(x, \dot{x}))\dot{\tilde{x}} + K_d(t)\tilde{x} = F_{ext} \tag{2.39}$$

As already shown in Section 2.3.1, a variable stiffness matrix leads to a loss of passivity. In fact, considering the positive definite storage function

$$\mathcal{V}(\tilde{x}, \dot{\tilde{x}}) = \frac{1}{2}\dot{\tilde{x}}^T\Lambda(x)\dot{\tilde{x}} + \frac{1}{2}\tilde{x}^T K_d(t)\tilde{x} \tag{2.40}$$

it follows that

$$\dot{\mathcal{V}} = \dot{\tilde{x}}^T\Lambda(x)\ddot{\tilde{x}} + \frac{1}{2}\dot{\tilde{x}}^T\dot{\Lambda}(x)\dot{\tilde{x}} + \tilde{x}^T K_d(t)\dot{\tilde{x}} + \frac{1}{2}\tilde{x}^T\dot{K}_d(t)\tilde{x} \tag{2.41}$$

Making the same computations that led from (2.38) to (2.17) one can obtain

$$\dot{\mathcal{V}} = \dot{\tilde{x}}^T F_{ext} + \left[\frac{1}{2}\tilde{x}^T \dot{K}_d(t)\tilde{x} - \dot{\tilde{x}}^T D_d \dot{\tilde{x}}\right] \tag{2.42}$$

Because of the stiffness variability, the sign of the term between brackets is undefined, the balance (2.18) holds no more and, in general, the system is not passive. In other words, the variation of the stiffness produces extra energy that is injected into the system destroying its passivity.

The loss of passivity severely affects the impedance control: a stable interaction is no more guaranteed and, in free motion, the tracking error $\tilde{x}(t)$ goes no more to zero.

As for the admittance control, in order to passively implement the stiffness variation, the impedance model is augmented with an energy tank, whose role is to store the energy dissipated by the controlled system.

The extended dynamics is then given by:

$$\begin{cases} \Lambda(x)\ddot{\tilde{x}} + (D_d + \mu(x,\dot{x}))\dot{\tilde{x}} + K_c\tilde{x} - wx_t = F_{ext} \\ \dot{x}_t = \dfrac{\sigma}{x_t}(\dot{\tilde{x}}^T D_d \dot{\tilde{x}}) - w^T\dot{\tilde{x}} \\ y = \left(\dot{\tilde{x}}^T x_t\right)^T \end{cases} \tag{2.43}$$

where $K_d(t)$, $T(x_t)$ and σ are defined in (2.23), (2.5) and (2.7) respectively.

Considering the model of the robot in the workspace (2.11), it is easy to see that the augmented system reported in (2.43) can be obtained by setting

$$F_\tau = F_g(x) + \Lambda(x)\ddot{x}_d + \mu\left(x,\dot{x}\right)\dot{x}_d - D_d\dot{\tilde{x}} - K_c\tilde{x} + wx_t \tag{2.44}$$

Slightly extending this result, it is possible to obtain the input force vector F_τ corresponding to the dynamics in (2.14).

The energy stored in the tank can be exploited for implementing the variable stiffness behavior by setting:

$$w(t) = \begin{cases} -\dfrac{K_v(t)\tilde{x}}{x_t} & if\ T(x_t) > \varepsilon \\ 0 & otherwise \end{cases} \tag{2.45}$$

where $\varepsilon > 0$ is a threshold below which energy cannot be extracted by the tank for avoiding singularities in (2.43). If there is enough energy in the tank, then replacing (2.45) in (2.43) gives the desired variable stiffness. If $T(x_t) \leq \varepsilon$, then the desired variable stiffness cannot be implemented passively and the constant stiffness K_c is implemented.

Passivity of the system (2.43) can be proven following the same considerations of Prop. 1. Using the augmented control structure reported in (2.43), it is possible to safely interact with any passive environment while implementing a variable stiffness in

the impedance model. Furthermore, when the system is not excited (i.e. $F_{ext} = 0$ and $K_v(t) = const$), the mechanical subsystem in (2.43) is characterized by a constant stiffness and, using the same procedure shown above, it can be shown that $\tilde{x}(t) \mapsto 0$ and that, therefore, at steady state the energy requested from the tank is zero.

The variation of stiffness that can be implemented using (2.45) is the best among those compatible with the passivity of the controlled system, namely with stable interactions which is a central requirement in impedance control.

2.4 Switch from autonomous to teleoperated mode

The problem of safe switching from autonomous to teleoperated mode will be addressed using energetic considerations and, therefore, it is convenient to formulate the problem in the port-Hamiltonian framework.

When switching from the autonomous mode to the teleoperated mode, there is a fully actuated surgical robot, the slave, that has to be teleoperated by means of a fully actuated mechanical interface, the master. In the following formulation will be considered bilateral teleoperation, where the teleoperated robot reflects back to the master reaction forces from the task being performed.

Master and slave robots can be modeled as port-Hamiltonian systems:

$$\begin{cases} \begin{pmatrix} \dot{x}_i \\ \dot{p}_i \end{pmatrix} = \begin{pmatrix} 0 & I \\ -I & -R_i \end{pmatrix} \begin{pmatrix} \frac{\partial H_i}{\partial x_i} \\ \frac{\partial H_i}{\partial p_i} \end{pmatrix} + \begin{pmatrix} 0 \\ I \end{pmatrix} F_{ext,i} + \begin{pmatrix} 0 \\ I \end{pmatrix} F_i \\ \quad y_i = \begin{pmatrix} 0 & I \end{pmatrix} \begin{pmatrix} \frac{\partial H_i}{\partial x_i} \\ \frac{\partial H_i}{\partial p_i} \end{pmatrix} = v_i \qquad i = m, s \end{cases} \tag{2.46}$$

where $x_i \in \mathbb{R}^n$, $p_i \in \mathbb{R}^n$ and $v_i \in \mathbb{R}^n$ represent the pose, the momentum and the velocity. H_i is the kinetic energy of the robot and R_i is a symmetric positive definite matrix representing the damping in the system, possibly augmented using local damping injection [40]. F_i is the generalized force due to the bilateral coupling, while $F_{ext,i}$ represents the force due to the interaction with the external world. For the master and the slave this force is indicated by F_h, the force applied by the human, and by F_e, the force applied by the environment, respectively. The surgeon is considered to be in the proximity of the surgical robot and, therefore, can be assumed a negligible communication delay between the master and the slave. Because of their robustness, indirect force feedback teleoperation systems have proven to be very effective in several applications and, therefore, will be considered this kind of coupling between master and slave. Several control strategies are available in the literature but one of the most simple, efficient and commonly used is a PD coupling

(see e.g. [48, 49]) that is described by:

$$\begin{cases} F_m = -K(x_m - x_s) - B(\dot{x}_m - \dot{x}_s) \\ F_s = +K(x_m - x_s) + B(\dot{x}_m - \dot{x}_s) \end{cases} \tag{2.47}$$

where $K \in \mathbb{R}^{n\times n} > 0$ and $B \in \mathbb{R}^{n\times n} > 0$ are the proportional and the derivative gains respectively. This is equivalent to interconnect master and slave with a (virtual) spring-damper system as illustrated in Fig. 2.3. Since the PD controller is a passive system and

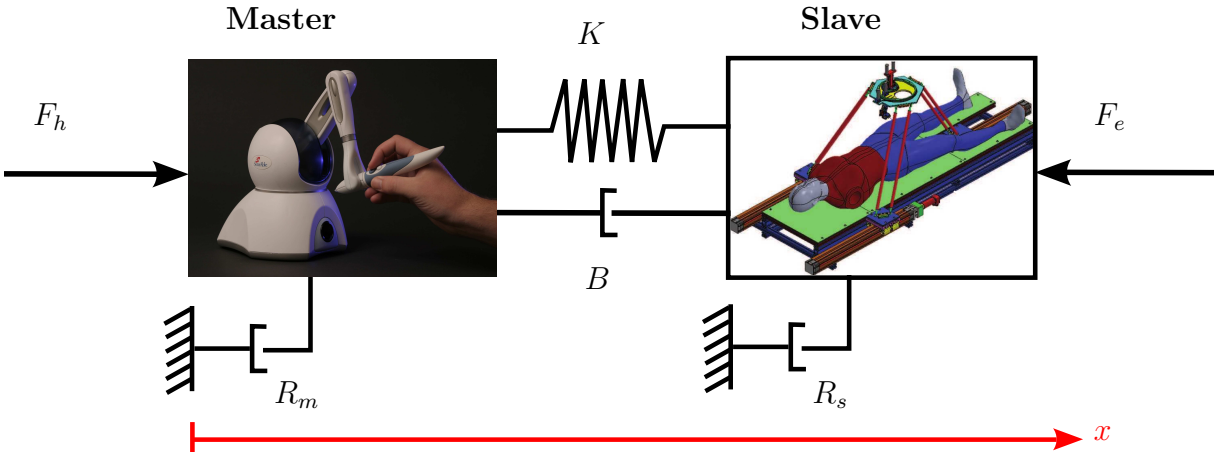


Figure 2.3: Master and slave are interconnected with a virtual spring-damper system equivalent to a PD coupling.

master and slave are passive port-Hamiltonian systems, the overall teleoperation system is passive and therefore, it is characterized by a stable behavior. Furthermore, in case master and slave are initially aligned, a PD coupling guarantees a steady state position tracking and in case of interaction with the environment a perfect steady state force tracking [40].

Nevertheless, in case of an initial position mismatch, a PD coupling may be problematic. In fact, suppose that at time t_s the surgical system switches from autonomous to teleoperated mode. If master and slave robots are still but misaligned, an abrupt force $K(x_m(t_s) - x_s(t_s))$ is applied, with different sign, to both robots. At the master side, this force provides an abrupt and, possibly, unexpected feedback to the user, while at the slave side, it generates a motion that may even be unacceptable, as is the case when the surgical tool is close to an organ. This abrupt force makes the control strategy (2.47) too dangerous to be used in surgical applications.

A simple but not too effective solution to this problem consists of two steps. First the pose of the master is taken to the same (scaled) pose of the slave and, second, the teleoperation system is started. The main advantage of this strategy is that master and slave positions match and using standard teleoperation strategy a zero steady state position error can be achieved. The main drawback of this approach is the waiting time

necessary for the alignment of the master. This waiting time cannot always be tolerable, especially in emergency situations. Furthermore, since positioning is made by standard PID controllers and an integral action on the position is needed for compensating the gravity effect, if the surgeon by accident (e.g. because of the rush due to the emergency) grasps and moves the master before the alignment is over, an unstable behaviors may arise [40] and the waiting time for activating the teleoperation system may increase.

In order to preserve the passivity of the overall teleoperation system and to get rid of abrupt forces due to the robots initial misalignment, (2.47) is modified as follows:

$$\begin{cases} F_m = -K(x_m - x_s - L) - B(\dot{x}_m - \dot{x}_s) \\ F_s = +K(x_m - x_s - L) + B(\dot{x}_m - \dot{x}_s) \end{cases} \tag{2.48}$$

where $L = x_m(t_s) - x_s(t_s)$ is the pose offset when the teleoperated mode is switched on. In this way, the coupling force transmitted to the robots is zero. Furthermore, (2.48) is still physically equivalent to a spring-damper system where the spring has a rest length L. It can be easily shown that this controller is passive and that, consequently, the overall teleoperation system is passive being the interconnection of passive systems. Thus, using the controller (2.48), when the teleoperated mode is switched on the surgeon can start using the system without waiting time and the system does not transmit any abrupt force during the switch on. However also this approach is not fully satisfactory because the surgeon has to mentally compensate for the position offset all the time and this may prevent her/him to focus on the primary task, namely emergency handling. Furthermore, the force/torque feedback received from the slave side is misleading due to the pose mismatch.

In the following will be shown how to exploit (2.48) for removing abrupt switching forces and how to design a passivity preserving control strategy generating a compensating force that reduces the waiting time with respect to the simplistic solution described above.

The steady state position error obtained when using (2.48) is due to the fact that the spring-like term of the controller has a non zero rest length L that, nevertheless, is necessary for preventing an abrupt force when the teleoperated mode is switched on.

The main idea for compensating the pose offset while avoiding abrupt initial forces is to gradually decrease the value of the rest length towards zero. In other words, the constant rest length L of (2.48) will be replaced by a continuous time function $l(t)$ such that $l(t_s) = L$ and $l(t) = 0$ for $t \geq t_s + t_c$. The time evolution of $l(t)$ and the amount of time $t_c - t_s$ necessary for completing the compensation are free parameters that can be set by the designer.

Nevertheless, since (2.48) is a bilateral interconnection, when the rest length of the spring-like element is changing, an elastic force is applied both to the master and to

the slave. This effect is undesired because the force due to the compensation will cause a motion of the slave that is not directly commanded by the surgeon and this is unacceptable since severe damages may be caused to the patient.

A possible solution can be to implement the following coupling between master and slave:

$$\begin{cases} F_{md} = -K(x_m - x_s - l(t)) - B(\dot{x}_m - \dot{x}_s) \\ F_{sd} = \alpha(t)[K(x_m - x_s - l(t)) + B(\dot{x}_m - \dot{x}_s)] \end{cases} \tag{2.49}$$

where F_{md} and F_{sd} indicate the desired values of F_m and F_s respectively. The smooth function $\alpha(t) : \mathbb{R} \mapsto [0, 1]$ is used for weighting the desired force to be applied to the slave side and it is defined by:

$$\alpha(t) = \begin{cases} \alpha_1(t)\alpha_2(t) & \text{if } t_s < t < t_M \\ 1 & \text{if } t \geq t_M \end{cases} \tag{2.50}$$

where t_M is the first instant of time at which $\alpha_1(t)\alpha_2(t) = 1$. Loosely speaking, when $\alpha(t)$ reaches its maximum value, it holds it. Let $e(t) = \|x_m(t) - x_s(t)\|$ be the norm of the position error and let $\lambda(t) = \|l(t)\|$. The map $\alpha_1 = \alpha_1(e(t)) : \mathbb{R}^+ \mapsto [0, 1]$ is a smooth real function defined as:

$$\alpha_1(e(t)) = \begin{cases} 1 & \text{if } e(t) \leq \bar{e}_1 \\ f_1(e(t)) & \text{if } \bar{e}_1 < e(t) < \bar{e}_2 \\ 0 & \text{if } e(t) \geq \bar{e}_2 \end{cases} \tag{2.51}$$

where $f_1(e(t))$ is a non increasing function and $\bar{e}_1 < \bar{e}_2$ are thresholds that can be set by the designer. Similarly, $\alpha_2 = \alpha_2(\lambda(t)) : \mathbb{R}^+ \mapsto [0, 1]$ is a smooth real function defined as:

$$\alpha_2(\lambda(t)) = \begin{cases} 1 & \text{if } \lambda(t) \leq \bar{\lambda}_1 \\ f_2(\lambda(t)) & \text{if } \bar{\lambda}_1 < \lambda(t) < \bar{\lambda}_2 \\ 0 & \text{if } \lambda(t) \geq \bar{\lambda}_2 \end{cases} \tag{2.52}$$

where $f_2(\lambda(t))$ is a non increasing function and $\bar{\lambda}_1 < \bar{\lambda}_2$ are thresholds that can be set by the designer. Functions α_1 and α_2 are used to weight the position error and the rest length. Only when both $e(t)$ and $\lambda(t)$ are small enough, $\alpha(t) = 1$ and (2.49) is equivalent to (2.47). In this way, a standard bilateral interconnection between master and slave will be established only when the position offset is close to zero (i.e. no compensating effort requested to the user) and when the position error is close to zero (i.e. no abrupt switching force). By properly choosing the thresholds in (2.51), (2.52), it is possible to tune the interconnection phase. Greater values of the thresholds will lead to a faster bilateral interconnection but in this case the surgeon will have to deal with spurious forces related to the final part of the rest length compensation.

The transient phase, when $0 < \alpha_1 < 1$ or $0 < \alpha_2 < 1$, can be made as short as desired, in order to mimic an almost discontinuous interconnection. If a more gradual connection

is desired, there will be a transient during which $0 < \alpha < 1$ and the control force to apply to the slave will be only partially implemented. This will provide the surgeon with a partial remote control capabilities that is made safe thanks to the scaling. In fact, the control force is scaled with a gain that is as smaller as the situation is "harder" to control (i.e. big position error and big rest length).

As long as $\alpha(t) = 0$, the slave is disconnected from the bilateral controller and no force is applied to the surgical robot. At the master side, the surgeon receives a force feedback due to the compensation. The feedback attracts the surgeon towards a pose of the master that is aligned with that of the slave, acting like a sort of virtual fixture. This makes the compensation process faster than a simple position controller since the user drives the master towards the desired pose and this action is superimposed to the one of the coupling controller.

Thus, using (2.49), the switching between automatic and teleoperated mode is smooth, a virtual fixture attracts the surgeon towards an aligned pose and the compensation process does not cause undesired motions of the slave robot. Nevertheless, it is well known that changing the rest length of a spring is not a passivity preserving operation (see e.g. [40]). Furthermore, unlike (2.48), the coupling reported in (2.49) is asymmetric and this asymmetry destroys the passivity of the controller. This loss of passivity makes the teleoperation system potentially unstable and unsafe to use.

In order to recover the passivity of the teleoperation system and to preserve the performance of (2.49), will be exploited the two-layer framework proposed in [26]. The overall architecture is reported in Fig. 2.4 and it can be decomposed into two layers: a *Transparency Layer* and a *Passivity Layer*.

First of all, master and slave robots are augmented with a tank, as described in Section 2.2, in order to have a storage of energy that can be used for implementing external desired control actions.

In the Transparency Layer master and slave exchange position and velocity information that is used for computing the desired coupling forces, namely F_{md} and F_{sd} defined in (2.49). These forces are sent to the Passivity Layer whose role is to passively implement them using the energy stored in the tanks. Master and slave energy tanks can exchange power for balancing the amount of energy stored at master and slave side.

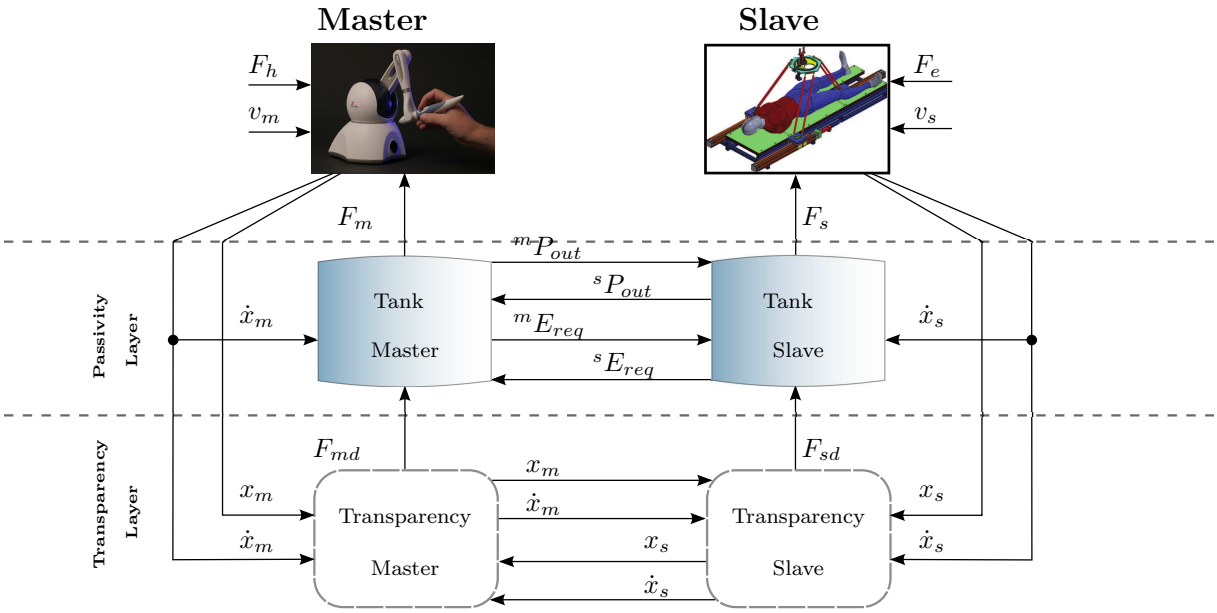


Figure 2.4: The two-layer architecture for the pose offset compensation. In the *Transparency Layer*, the desired coupling forces are computed according to (2.49). These commands are sent to the *Passivity Layer*, whose role is to check and guarantee the passivity of the total system.

Formally, the augmented master and slave sides are modeled as:

$$\begin{cases} \begin{pmatrix} \dot{x}_i \\ \dot{p}_i \end{pmatrix} = \begin{pmatrix} 0 & I \\ -I & -R_i \end{pmatrix} \begin{pmatrix} \frac{\partial H_i}{\partial x_i} \\ \frac{\partial H_i}{\partial p_i} \end{pmatrix} + \begin{pmatrix} 0 \\ I \end{pmatrix} F_{ext,i} + \begin{pmatrix} 0 \\ I \end{pmatrix} F_i \\ \dot{x}_{t_i} = \dfrac{\sigma_i}{x_{t_i}} D_i(x_i) + \dfrac{1}{x_{t_i}} (\sigma_i {}^iP_{in} - {}^iP_{out}) + u_{t_i} \\ y_i = \begin{pmatrix} v_i \\ y_{t_i} \end{pmatrix} \qquad i = m, s \end{cases} \tag{2.53}$$

where x_{t_i}, $y_{t_i} = x_{t_i}$ and $T_i = \frac{1}{2} x_{t_i}^2$ are the state of the tank, the output associated to the tank and the energy stored in the tank respectively. D_i is the energy dissipated by the robot (that can be augmented by introducing a local damping injection [42]) and ${}^iP_{in}$ and ${}^iP_{out}$ are the power flows that can be exchanged with the other tank.

The desired coupling forces are implemented using the energy stored in the tanks by interconnecting the power port of the tank (u_{t_i}, y_{t_i}) with the power port of the robot

(F_i, v_i) using the following power preserving interconnection:

$$\begin{cases} F_i = \dfrac{F_{id}}{x_{t_i}} y_{t_i} = \dfrac{F_{id}}{x_{t_i}} x_{t_i} = F_{id} \\ u_{t_i} = -\dfrac{F_{id}^T}{x_{t_i}} v_i \end{cases} \qquad i = m, s \tag{2.54}$$

As shown in (2.8) and (2.9), (2.54) is a power preserving interconnection and, therefore, the desired input F_{id} is implemented by exchanging energy with the tank: if $F_{id}^T v_i > 0$, it means that some energy is necessary for implementing the desired input and, consequently, this energy is extracted from the tank. On the other hand, if $F_{id}^T v_i < 0$ it means that the desired action is dissipative and, therefore, the dissipated energy is stored in the tank.

The desired coupling forces can be passively achieved if and only if tanks are full enough. As reported in Section 2.2, if the energy of the tank T_i goes below a predefined threshold ε_i, energy extraction is forbidden and the coupling force implemented is $F_i = 0$. This guarantees passivity and, therefore, a safe behavior of the teleoperation system, but prevents from achieving good performance.

Thus, it is necessary to ensure that, at both master and slave side, the filling level of the tank is kept quite higher than the minimum. The energy stored in the tanks can be augmented in two ways: receiving energy from the other tank or augmenting the damping of the associated robot by means of damping injection. The latter option should be used as a last resort since it is an invasive option and the extra damping will be felt by the user and/or will modify the dynamics of the slave changing the desired coupling (2.49).

An exchange of energy between master's and slave's tanks affects only the amount of energy available for implementing a desired control action and it does not have any direct effect on the dynamics of the teleoperation system. Following the approach suggested in [50], will be implemented the following exchange strategy: when the energy of the tank of the master goes below a desired threshold ${}^mT_{req}$, an energy request ${}^mE_{req}$ is sent to the slave tank. If the amount of energy available in the tank of the slave is greater than a threshold ${}^sT_{ava}$, a flow of energy is sent to the master tank. Furthermore, the slave tank sends to the master tank also the energy dissipated by the robot that cannot be stored in the tank because it reached its upper threshold $\bar{T}_s$. An analogous behavior is implemented at the slave side.

Formally, we have that the energy request signal is given by:

$${}^iE_{req} = \begin{cases} 1 & if\ T_i(x_{t_i}) < {}^iT_{req} \\ 0 & otherwise \end{cases} \qquad i = m, s \tag{2.55}$$

The overall power extracted from each tank and sent to the other tank is then given by:

$$\begin{cases} {}^mP_{out} = (1 - \sigma_m) D_m + {}^sE_{req} \beta_m \bar{P} = {}^sP_{in} \\ {}^sP_{out} = (1 - \sigma_s) D_s + {}^mE_{req} \beta_s \bar{P} = {}^mP_{in} \end{cases} \tag{2.56}$$

where the energy dissipated by the robot is sent to the other tank if and only if the tank is already full (i.e. $\sigma_i = 1$). The variable $\beta_i \in \{0, 1\}$ disables/enables the transfer of the energy stored in the tank and it is given by:

$$\beta_i = \begin{cases} 1 & if\ T_i(x_{t_i}) \geq {}^iT_{ava} \\ 0 & otherwise \end{cases} \qquad i = m, s \tag{2.57}$$

The variable $\bar{P} > 0$ is the rate of energy flowing from one tank to the other and it is a design parameter. The bigger is $\bar{P}$, the faster is the energy transfer. All the thresholds mentioned so far are also design parameters and the following constraints must be satisfied: $\varepsilon_i < {}^iT_{req} < {}^iT_{ava} < \bar{T}_i$, for $i = m, s$.

The strategy illustrated so far guarantees the passivity of the teleoperation system as proven in the following.

Proposition 2. *The overall teleoperation system is passive with respect to the pair*

$$((F_h^T, F_e^T)^T, (v_m^T, v_s^T)^T)$$

Proof. Consider the total energy of the teleoperation system as a storage function:

$$\mathcal{H}(t) = H_m(t) + H_s(t) + T_m(t) + T_s(t) \tag{2.58}$$

Using (2.53) it follows that

$$\begin{aligned} \dot{\mathcal{H}}(t) = \ & -D_m(t) - D_s(t) + F_m^T v_m + F_s^T v_s + \sigma_m D_m(t) + \sigma_m {}^mP_{in} - {}^mP_{out} + \\ & + \sigma_s D_s(t) + \sigma_s {}^sP_{in} - {}^sP_{out} + u_{t_m} y_{t_m} + u_{t_s} y_{t_s} + F_h^T v_m + F_e^T v_s \end{aligned} \tag{2.59}$$

Since the tanks and the robots are interconnected by means of the power preserving interconnection (2.54), it follows that $F_i^T v_i = -u_{t_i} y_{t_i}$ where $i = m, s$ and therefore (2.59) can be rewritten as:

$$\begin{aligned} \dot{\mathcal{H}}(t) = \ & -(1-\sigma_m) D_m(t) - (1-\sigma_s) D_s(t) + \sigma_m {}^mP_{in} - {}^mP_{out} + \\ & + \sigma_s {}^sP_{in} - {}^sP_{out} + F_h^T v_m + F_e^T v_s \end{aligned} \tag{2.60}$$

Using (2.56), one can obtain

$$\begin{aligned} \dot{\mathcal{H}}(t) = \ & -(1-\sigma_m) D_m(t) - (1-\sigma_s) D_s(t) - (1-\sigma_m) {}^sP_{out} - \\ & - (1-\sigma_s) {}^mP_{out} + F_h^T v_m + F_e^T v_s \end{aligned} \tag{2.61}$$

Since $\sigma_i \in \{0, 1\}$ and since ${}^iP_{out}, D_i \geq 0$ for $i = m, s$, it follows that:

$$\dot{\mathcal{H}}(t) \leq F_h^T v_m + F_e^T v_s \tag{2.62}$$

which proves the statement. □

Passivity guarantees a safe behavior of the teleoperation system during the compensation phase and while teleoperating the slave.

If the exchange of energy between the tanks is not sufficient for guaranteeing a sufficient level of energy in the tanks, then it is necessary to augment the local damping of the robots. Intuitively, increasing the damping is a passivity preserving operation and, formally, passivity can be proven following the same lines of Prop. 2, using a variable damping matrix in the port-Hamiltonian models of master and slave.

2.5 Bilateral teleoperation for a dual arms manipulator with virtual fixtures

As introduced in Section 2.1, teleoperating a multi-arm robot is a complex task that surgeons are able to accomplish only after a long training. Virtual fixtures and potential fields are usually exploited for assisting the surgeon during teleoperation of such complex systems. However, when an attractive fixture is exploited for driving the user towards a certain path, it can happen that the repulsive forces generated by the fixtures devoted to collision avoidance take the operator far from the desired path leading to an inefficient task execution. In this section, a novel strategy to improve the efficiency of the teleoperation system, will be introduced. The strategy will allow to change the direction of the assistive force generated by the virtual fixture while preserving the passivity of the overall system.

A teleoperation system for a dual arms robotic manipulator will be considered, as shown in Fig. 2.5. The master side consists of a pair of haptic devices. Each device can be used to teleoperate one arm of the dual robotic system. Each arm of the dual robotic system, namely the slave side, is a n-dimensional fully actuated robotic manipulator. In order to obtain force feedback while preserving robustness and safety of the overall teleoperation system, will be assumed that each haptic device and each robot is controlled by means of an impedance/admittance controller and that it behaves as a passive, decoupled mechanical system [51]. Furthermore will be assumed that each device is coupled with the corresponding arm by means of a passive bilateral coupling (e.g. position-position architecture [52], [40]). Since the motivation of this thesis is surgical automation and teleoperation and since in most of the cases all the robots are in the same room, a negligible communication delay between master and slave will be assumed.

In order to assist the operator during the teleoperation, virtual fixtures are exploited and virtual forces are generated on the master side. The environment at the slave side is mapped, possibly after scaling, to the master side and that the fixtures can be computed locally. This mapping can be done exploiting exteroceptive sensors (e.g. cameras) usually

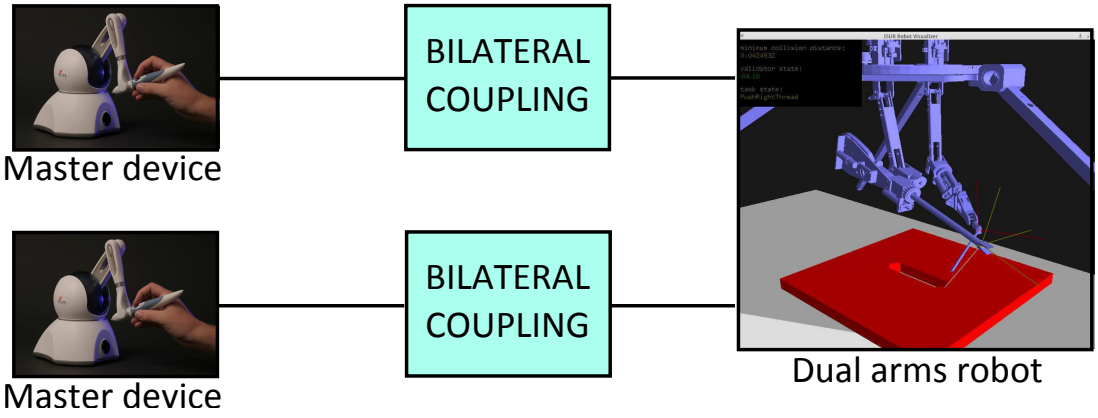


Figure 2.5: The bilateral teleoperation system for the dual arms robotic device.

mounted on surgical robots for providing situational awareness to the surgeon. In this way, the virtual force due to the fixtures are felt clearly and directly by the user that can promptly react and change its plans unlike in the case where virtual fixtures are implemented on the slave side, where the fixture forces arrive to the user filtered and degraded by the bilateral coupling. Furthermore, by properly tuning the bilateral controllers coupling master and slave, the (scaled) position error between master and slave can be made very small. Unfortunately reflected forces are much less accurate. As proposed in [53], fixture will generate translational forces only.

The teleoperation system, can be used in two modalities: the *free mode* and the *assisted mode*. In free mode, the operator can drive the robots with no further assistance and virtual fixtures are exploited only for obstacle avoidance. A repulsive potential surrounds each (mapped) obstacle (e.g. organs) and a repulsive force is applied to the haptic device when the user gets closer to a dangerous area. In the assisted mode, virtual fixtures are exploited for helping the user in the execution of a task (e.g. suturing a wound). A collision free reference path for the task is generated and a virtual fixture attracting the master toward such a path is produced. While the operator is approaching the reference path it may encounter an obstacle with one or both the arms. Thus, it may happen that the repulsive force generated by the fixture drives the arms in an inconvenient configuration and that the operator has to spend some time trying to take back the robots in the right position. This inefficient behavior is due to the fact that virtual fixtures are designed independently of each other and it may be very undesirable in emergency situations, when time is really precious. Thus, during the assisted mode a time variable rotation of the repulsive virtual force is implemented in order to direct as much as possible total fixture force towards the reference path, taking the surgeon as fast as possible towards the task that has to be executed making the overall behavior more efficient.

In order to preserve the safe and stable behavior guaranteed by a passivity based teleoperation system, it is necessary to prove that the introduction of virtual fixtures and the rotation of virtual forces does not threaten the passivity of the overall teleoperation system. In the following will be formally shown how to embed virtual fixtures at the master side on a passivity based bilateral teleoperation system while preserving the passivity of the overall system. Furthermore, a condition under which a rotation of the virtual force will preserve the passivity of the overall system will be derived. Moreover, attractive and repulsive fixture for the dual robotic system will be designed and an algorithm for choosing the amount of the rotation of the repulsive virtual forces will be provided.

2.5.1 Free mode

For the passivity analysis only one haptic device-robot pair will be considered. All the results can be applied verbatim to the other pair.

Since both the haptic device and the robot are impedance controlled in order to behave as decoupled mechanical systems and since virtual fixtures will generate translational forces only, as suggested by [53], to keep the analysis simple, without loss of generality the focus will be on the translational DOFs only. As in Section 2.4, master and slave robots can be modeled as port-Hamiltonian systems:

$$\begin{cases} \begin{pmatrix} \dot{x}_i \\ \dot{p}_i \end{pmatrix} = \begin{pmatrix} 0 & I \\ -I & -D_i \end{pmatrix} \begin{pmatrix} \frac{\partial H_i}{\partial x_i} \\ \frac{\partial H_i}{\partial p_i} \end{pmatrix} + \begin{pmatrix} 0 \\ I \end{pmatrix} F_{ext,i} + \begin{pmatrix} 0 \\ I \end{pmatrix} F_i + \begin{pmatrix} 0 \\ I \end{pmatrix} F_{ast,i} \\ y_i = \begin{pmatrix} 0 & I \end{pmatrix} \begin{pmatrix} \frac{\partial H_i}{\partial x_i} \\ \frac{\partial H_i}{\partial p_i} \end{pmatrix} = v_i \qquad i = m, s \end{cases} \tag{2.63}$$

where $x_i \in \mathbb{R}^3$, $p_i \in \mathbb{R}^3$ and $v_i \in \mathbb{R}^3$ represent the pose, the momentum and the velocity. H_i is the kinetic energy of the robot and D_i is a symmetric positive definite matrix representing the damping in the system. F_i is the force due to the bilateral coupling, while $F_{ext,i}$ represents the force due to the interaction with the external world. For the master and the slave this force is indicated by F_h, the force applied by the human, and by F_e, the force applied by the environment, respectively. Finally, $F_{ast,i}$ is the force due to the virtual fixtures.

In free mode, the master moves in a potential field given by the sum of potential fields generating a repulsive force from an obstacle.

$$V(x_m) = \sum_{i=1}^{p} V_{rep}(x_m) \tag{2.64}$$

and the force due to the potential field is given by

$$F_{ast,i} = \begin{cases} -\frac{\partial V}{\partial x_m} & \text{if } i = m \\ 0 & \text{if } i = s \end{cases} \tag{2.65}$$

A very effective and commonly used bilateral passive coupling is the following PD dynamics [48], [49]:

$$\begin{cases} \dot{x}_c = v_m - v_s = v_c \\ F_c = K(x_m - x_s) + B(\dot{x}_m - \dot{x}_s) \end{cases} \tag{2.66}$$

where $K \in \mathbb{R}^{3\times3} > 0$ and $B \in \mathbb{R}^{3\times3} > 0$ are the proportional and the derivative gains respectively. This is equivalent to interconnect master and slave with a (virtual) spring-damper system where

$$\begin{cases} F_m = -F_c \\ F_s = F_c \end{cases} \tag{2.67}$$

Furthermore, let $H_c(t) = \frac{1}{2}x_c^T K x_c$ be the energy of the virtual spring. It can be easily seen that the following balance holds:

$$F_c^T v_c \geq \dot{H}_c \tag{2.68}$$

Proposition 3. *The teleoperation system with the virtual fixtures embedded at the master side is passive with respect to the the pair* $((F_h^T, F_e^T)^T, (v_m^T, v_s^T)^T)$.

Proof. Since the master is a port-Hamiltonian system, using (2.2), one can obtain

$$F_h^T v_m + F_m^T v_m + F_{ast,m}^T v_m \geq \dot{H}_m \tag{2.69}$$

Using (2.79) in (2.69) gives

$$F_h^T v_m + F_m^T v_m - \frac{\partial^T V}{\partial x_m} v_m \geq \dot{H}_m \tag{2.70}$$

and, consequently

$$F_h^T v_m + F_m^T v_m \geq \dot{H}_m + \frac{\partial^T V}{\partial x_m}\dot{x}_m = \dot{H}_m + \dot{V} = \dot{\tilde{H}}_m \tag{2.71}$$

Applying (2.2) to the slave it follows that:

$$F_s^T v_s + F_e^T v_s \geq \dot{H}_s \tag{2.72}$$

Thus, summing (2.71), (2.72) and (2.68) yields:

$$F_h^T v_m + F_m^T v_m + F_c^T v_c + F_s^T v_s + F_e^T v_s \geq \dot{\tilde{H}}_m + \dot{H}_c + \dot{H}_s \tag{2.73}$$

Considering (2.66) and (2.47), it follows that

$$F_c^T v_c = F_c^T v_m - F_c^T v_s = -F_m^T v_m - F_s^T v_s \tag{2.74}$$

Using (2.74) in (2.73) and considering the lower bounded storage function

$$\mathcal{H}(t) = \bar{H}_m(t) + H_c(t) + H_s(t) \tag{2.75}$$

it follows that:

$$F_h^T v_m + F_m^T v_m - F_m^T v_m - F_s^T v_s + F_s^T v_s + F_e^T v_s \geq \dot{\mathcal{H}} \tag{2.76}$$

and, consequently

$$F_h^T v_m + F_e^T v_s \geq \dot{\mathcal{H}}(t) \tag{2.77}$$

which proves the statement. □

Remark 3. *The result of Prop. 3 can be easily extended for coping with any passive bilateral interconnection between master and slave.*

2.5.2 Assisted mode

In the assisted mode, the master moves in a potential field given by the sum of potential fields generating a repulsive force from an obstacle and of a potential field generating a virtual force attracting the master towards a reference trajectory.

$$V(x_m) = V_{att}(x_m) + \sum_{i=1}^{p} V_{rep}(x_m) \tag{2.78}$$

The force due to the potential field is given by

$$F_{ast,i} = \begin{cases} -\frac{\partial V}{\partial x_m} & \text{if } i = m \\ 0 & \text{if } i = s \end{cases} \tag{2.79}$$

As described above, standard virtual forces could point the operator along inefficient directions. To make the virtual fixture more efficient will be shown how to rotate the virtual force for pointing it towards the best task dependent direction. Formally, a rotation matrix $R(t) \in SO(3)$ is introduced, by means of which the direction of the assistive force can be changed.

Of course the rotation of the virtual force has to be a passivity preserving operation in order to keep on guaranteeing a stable behavior of the overall system. Let θ be the angle between the $\dot{x}_m \in \mathbb{R}^3$ and the assistive force $\frac{\partial V}{\partial x_m}$ and let θ_R be the angle between the vector $\dot{x}_m$ and the rotated force $R(t)\frac{\partial V}{\partial x_m}$. Then, the following proposition allows to define the rotations that could be performed without producing extra energy and satisfying the passivity condition.

Proposition 4. *The teleoperation system with the rotated virtual fixtures embedded at the master side is passive with respect to the the pair* $((F_h^T, F_e^T)^T, (v_m^T, v_s^T)^T)$ *if the following constraint is satisfied:*

$$cos(\theta) \leq cos(\theta_R) \tag{2.80}$$

Proof. If the rotation R is applied to the assistive force, the balance in (2.69) becomes

$$F_h^T v_m + F_m^T v_m + F_{ast,m}^T R^T v_m \geq \dot{H}_m \tag{2.81}$$

and thus

$$F_h^T v_m + F_m^T v_m - \frac{\partial^T V}{\partial x_m} R^T v_m \geq \dot{H}_m \tag{2.82}$$

$$F_h^T v_m + F_m^T v_m \geq \dot{H}_m + \frac{\partial^T V}{\partial x_m} R^T \dot{x}_m \tag{2.83}$$

According to (2.71), taking

$$\frac{\partial^T V}{\partial x_m} R^T \dot{x}_m \geq \frac{\partial^T V}{\partial x_m} \dot{x}_m \tag{2.84}$$

one can obtain that

$$F_h^T v_m + F_m^T v_m \geq \dot{H}_m + \frac{\partial^T V}{\partial x_m} R^T \dot{x}_m \geq \dot{H}_m + \dot{V} \tag{2.85}$$

and the same computations that bring from (2.68) to (2.77) can be applied to prove the passivity.

The two terms of the inequality (2.84) can be computed as scalar products as follows:

$$\begin{aligned} < \frac{\partial^T V}{\partial x_m} R^T, \dot{x}_m > &= \left\| \frac{\partial^T V}{\partial x_m} R^T \right\| \|\dot{x}_m\| \, cos(\theta_R) \\ < \frac{\partial^T V}{\partial x_m}, \dot{x}_m > &= \left\| \frac{\partial^T V}{\partial x_m} \right\| \|\dot{x}_m\| \, cos(\theta) \end{aligned} \tag{2.86}$$

Since the rotation of a vector doesn't change the value of it's norm, the first scalar product becomes:

$$< \frac{\partial^T V}{\partial x_m} R^T, \dot{x}_m >= \left\| \frac{\partial^T V}{\partial x_m} \right\| \|\dot{x}_m\| \, cos(\theta_R) \tag{2.87}$$

Then, substituting the scalar products into (2.84) one can obtain

$$\left\| \frac{\partial^T V}{\partial x_m} \right\| \|\dot{x}_m\| \, cos(\theta_R) \geq \left\| \frac{\partial^T V}{\partial x_m} \right\| \|\dot{x}_m\| \, cos(\theta) \tag{2.88}$$

and thus

$$cos(\theta) \leq cos(\theta_R) \tag{2.89}$$

□

Remark 4. *Since θ and θ_R are angles between two vectors, they belong to the interval $[0, \pi]$. To satisfy the constraint* (2.80) *it must be $\theta_R \leq \theta$.*

The angles θ and θ_R can be computed thanks to the following geometric considerations and to the axis-angle representation [54]. All the frames described in the following have their origin on the end-effector of the master robot and are referred to the world frame $\mathcal{F}_0$. The same considerations can be applied to both the master robots.

Let $\mathcal{F}_1$ be the frame defined with

- The z_1 axis expressed in the world frame ($\mathcal{F}_0$) taken as the orthogonal vector between $\dot{x}_m$ and $\frac{\partial V}{\partial x_m}$
$$\dot{x}_m \times \frac{\partial V}{\partial x_m} = {}^0n_{1z} \rightarrow {}^0\hat{z}_1 = \frac{{}^0n_{1z}}{\|{}^0n_{1z}\|} \tag{2.90}$$
- The y_1 axis expressed in the world frame taken along the direction of $\dot{x}_m$
$${}^0\hat{y}_1 = \frac{\dot{x}_m}{\|\dot{x}_m\|} \tag{2.91}$$
- The x_1 axis expressed in the world frame being the vector orthogonal to ${}^0n_{1z}$ and $\dot{x}_m$ and chosen following the right hand thumb rule
$${}^0n_{1z} \times \dot{x}_m = {}^0n_{1x} \rightarrow {}^0\hat{x}_1 = \frac{{}^0n_{1x}}{\|{}^0n_{1x}\|} \tag{2.92}$$

The unit vectors ${}^0\hat{x}_1$, ${}^0\hat{y}_1$ and ${}^0\hat{z}_1$ can be combined to obtain the rotation matrix that expresses the orientation of $\mathcal{F}_1$ with respect to $\mathcal{F}_0$:

$${}^0R_1 = \begin{bmatrix} {}^0\hat{x}_1 & {}^0\hat{y}_1 & {}^0\hat{z}_1 \end{bmatrix} \tag{2.93}$$

Then, consider a second frame $\mathcal{F}_2$ defined with the same z-axis as $\mathcal{F}_1$, and y_2 and x_2 defined as

$${}^0\hat{y}_2 = \frac{\frac{\partial V}{\partial x_m}}{\|\frac{\partial V}{\partial x_m}\|} \tag{2.94}$$

$${}^0n_{1z} \times \frac{\partial V}{\partial x_m} = {}^0n_{2x} \rightarrow {}^0\hat{x}_2 = \frac{{}^0n_{2x}}{\|{}^0n_{2x}\|} \tag{2.95}$$

The unit vectors ${}^0\hat{x}_2$, ${}^0\hat{y}_2$ and ${}^0\hat{z}_2$ can be combined to obtain the rotation matrix that expresses the orientation of $\mathcal{F}_2$ with respect to $\mathcal{F}_0$:

$${}^0R_2 = \begin{bmatrix} {}^0\hat{x}_2 & {}^0\hat{y}_2 & {}^0\hat{z}_2 \end{bmatrix} \tag{2.96}$$

By combining the rotation matrices ${}^0R_1$ and ${}^0R_2$ we can obtain the rotation matrix that rotate the vector $\dot{x}_m$ into the vector $\frac{\partial V}{\partial x_m}$ as:

$$\tilde{R} = {}^2R_1 = {}^2R_0\,{}^0R_1 = \left({}^0R_2\right)^T {}^0R_1 \tag{2.97}$$

The rotation matrix $\tilde{R}$ describes the rotation of an angle θ around the axis ${}^0n_{1z}$. The angle θ is given by:

$$\theta = arccos\left(\frac{\tilde{r}_{11} + \tilde{r}_{22} + \tilde{r}_{33} - 1}{2}\right) \tag{2.98}$$

and represents the angle between the vectors $\dot{x}_m$ and $\frac{\partial V}{\partial x_m}$ that has to be used in (2.80).

Now, the idea is to apply the repulsive force towards another direction. This means that a freely chosen rotation R is applied to the potential $\frac{\partial V}{\partial x_m}$.

With same considerations as before, one can define $\mathcal{F}_3$ and $\mathcal{F}_4$ and find the rotation matrix $\hat{R}$ that rotate the vector $\dot{x}_m$ into the vector $R\frac{\partial V}{\partial x_m}$. Indeed, by combining the matrices ${}^0R_3$ and ${}^0R_4$, i.e. matrices expressing the orientation of $\mathcal{F}_3$ and $\mathcal{F}_4$ with respect to $\mathcal{F}_0$, one can obtain

$$\hat{R} = {}^4R_3 = {}^4R_0\,{}^0R_3 = \left({}^0R_4\right)^T\,{}^0R_3 \tag{2.99}$$

The rotation matrix $\hat{R}$ describes the rotation of an angle θ_R around the axis ${}^0n_{3z}$. The angle θ_R is given by:

$$\theta_R = arccos\left(\frac{\hat{r}_{11} + \hat{r}_{22} + \hat{r}_{33} - 1}{2}\right) \tag{2.100}$$

and represents the angle between the vectors $\dot{x}_m$ and $R\frac{\partial V}{\partial x_m}$ that has to be used in (2.80). Figure 2.6 shows the representation of the frames described above.

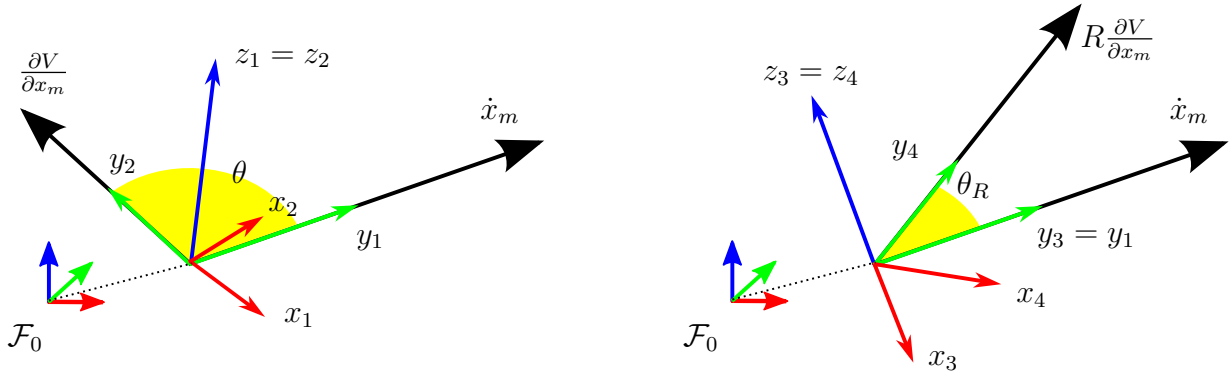


Figure 2.6: Representation of the different frames. The figure on the left shows frames $\mathcal{F}_1$ and $\mathcal{F}_2$, while the figure on the right shows frames $\mathcal{F}_3$ and $\mathcal{F}_4$.

2.5.3 Design of the virtual fixtures

In the assisted mode, the two arms of the manipulator are attracted towards two paths φ_A and φ_B computed, e.g, by the motion planner. Since the planned path is collision-free only if the motion of the two robots is simultaneous and synchronized by the progress of a single parameterizing scalar s, it is not possible to consider the two paths independently in the generation of the virtual fixtures for each arm.

To calculate the target points that must be attractive for the teleoperation masters, taking into account the synchronization between the two paths, it is necessary to find the optimal value of $\hat{s}$ and the corresponding path poses $\hat{t}_A = \varphi_A(\hat{s})$ and $\hat{t}_B = \varphi_B(\hat{s})$ such that the total distance among the end-effectors and their desired poses is minimized. The optimization problem is therefore setup as follows:

$$\hat{s} = argmin_{s\in[0,1]} \|\varphi_A(s) - x_A\| + \|\varphi_B(s) - x_B\| \tag{2.101}$$

where x_A and x_B indicate the poses of the two haptic devices.

The attractive potential fields $V_{att}(x_A)$ and $V_{att}(x_B)$ for the two haptic devices are then given by

$$V_{att}(x_A) = \tfrac{1}{2} k_p \left(x_A - \hat{t}_A\right)^2 \tag{2.102}$$

$$V_{att}(x_B) = \tfrac{1}{2} k_p \left(x_B - \hat{t}_B\right)^2 \tag{2.103}$$

where k_p is a position gain and $\hat{t}_A = \varphi_A(\hat{s})$ and $\hat{t}_B = \varphi_B(\hat{s})$ are computed from the result of (2.101).

The corresponding attractive force is defined as the negative gradient of the potential function, expressed as:

$$F_{att}(x_A) = -k_p(x_A - \hat{t}_A) \tag{2.104}$$

$$F_{att}(x_B) = -k_p(x_B - \hat{t}_B) \tag{2.105}$$

The attractive forces $F_{att}(x_A)$ and $F_{att}(x_B)$ allow the points x_A and x_B of the end-effectors to follow the reference paths.

The guidance fixture alone, even if designed to assist the user to move on a previously planned collision-free path, does not guarantee the actual absence of collision. Therefore, an additional artificial potential field $V_{rep}(x)$ is included to create a potential barrier around the obstacles or around specified forbidden regions. Since in this case the potential field is equal for both arms, x_A and x_B will be considered as a general pose x.

A classical expression for a repulsive potential function is:

$$V_{rep}(x) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho(x)} - \frac{1}{\rho_0}\right)^2 & if\ \rho(x) \le \rho_0 \\ 0 & if\ \rho(x) > \rho_0 \end{cases} \tag{2.106}$$

where η is a positive constant gain, ρ_0 represents the limit distance of the potential field (beyond this distance the potential do not affect the master's control) and $\rho(x)$ is the Euclidean distance from the nearest obstacle.

The corresponding repulsive force is defined as the negative gradient of the potential function, expressed as:

$$F_{rep}(x) = \begin{cases} \eta\left(\frac{1}{\rho(x)} - \frac{1}{\rho_0}\right)\left(\frac{1}{\rho^2(x)}\right)\left(\frac{\delta\rho(x)}{\delta x}\right) & if\ \rho(x) \le \rho_0 \\ 0 & if\ \rho(x) > \rho_0 \end{cases} \tag{2.107}$$

Optimization of the direction of the total assistive force. When the user guides the master out of the planned path, it may enter within the space in which the effect of the repulsive potential field is applied, pushing the robot away from the obstacle. Consider the repulsive potential field as a spherical bounding box Γ with a center $C = [x_c, y_c, z_c]$, assumed to be the center of the obstacle, and a radius ρ_0 which is the maximum region of influence of the potential field. Let $P_A = [x_p, y_p, z_p]$ be one of the infinite points on the sphere where the end-effector of the master A (same considerations can be applied to the master B) starts feeling the effect of the repulsive potential field.

The direction of the repulsive force is along the segment connecting C and P_A, so that it is normal to the surface of the sphere Γ. The total assistive force F_{ast} felt by the user is the resultant of the attractive force F_{att} and the repulsive force F_{rep}. If the latter is prevailing over the attractive one, which is a reasonable design choice if priority is given to obstacle avoidance, it may happen that the total assistive force pushes the master robot away from the desired point on the planned path, which is instead not desirable. Since the total assistive force can be rotated towards a desired direction without violating the passivity condition, if the constraint (2.80) is satisfied, this result can be exploited to redirect the assistive force in order to indicate a preferred obstacle avoidance direction, namely towards the planned path, even in case of a high risk of collision (i.e. the user has moved the master further inside the sphere Γ).

The examples shown in Fig. 2.7 consider two different cases. If the master is in P_{A1}, the assistive force F_{ast} would not move the robot towards to the reference on the path. Given the shown direction of $\dot{x}_{mA1}$, the rotations that satisfy the constraint in (2.80) (i.e. the possible rotation of the assistive force that would not violate the passivity condition) are those that keep the rotated vector RF_{ast} inside the cone defined by the altitude $\dot{x}_{mA1}$, the angle θ and the slant height F_{ast}, as shown in orange in Fig. 2.7. The optimal rotation of the assistive force in the direction of the planned path is the one that brings the rotated force to be the apothem of the cone that lies on the plane defined by the vector $\dot{x}_{mA1}$ and the segment $\overline{P_{A1}\hat{t}_A}$ (i.e. the *line-of-sight* from the master to the path). Of course, in a case in which the latter segment is inside the cone, the optimal rotation would redirect the assistive force exactly as its attractive component, which would amplify the push towards the path.

A different approach must be applied if the master is in a point P_{A2} on the opposite side of the sphere Γ with respect to the reference path, which is a condition that can be detected checking if the distance between C and the *line-of-sight* from the master to the path is less than the distance between C and the master. In this case, the cone described above would allow rotations pointing the assistive force to intersect the sphere, and thus

further inside the repulsive region. In such a case the rotated force must remain tangent to the sphere in P_{A2}. However, since in general there are infinite tangent vectors satisfying the passivity constraint, the optimal direction will be considered as the one that points as much as possible towards the projection of $\hat{t}_A$ on the plane tangent to the sphere Γ in P_{A2}.

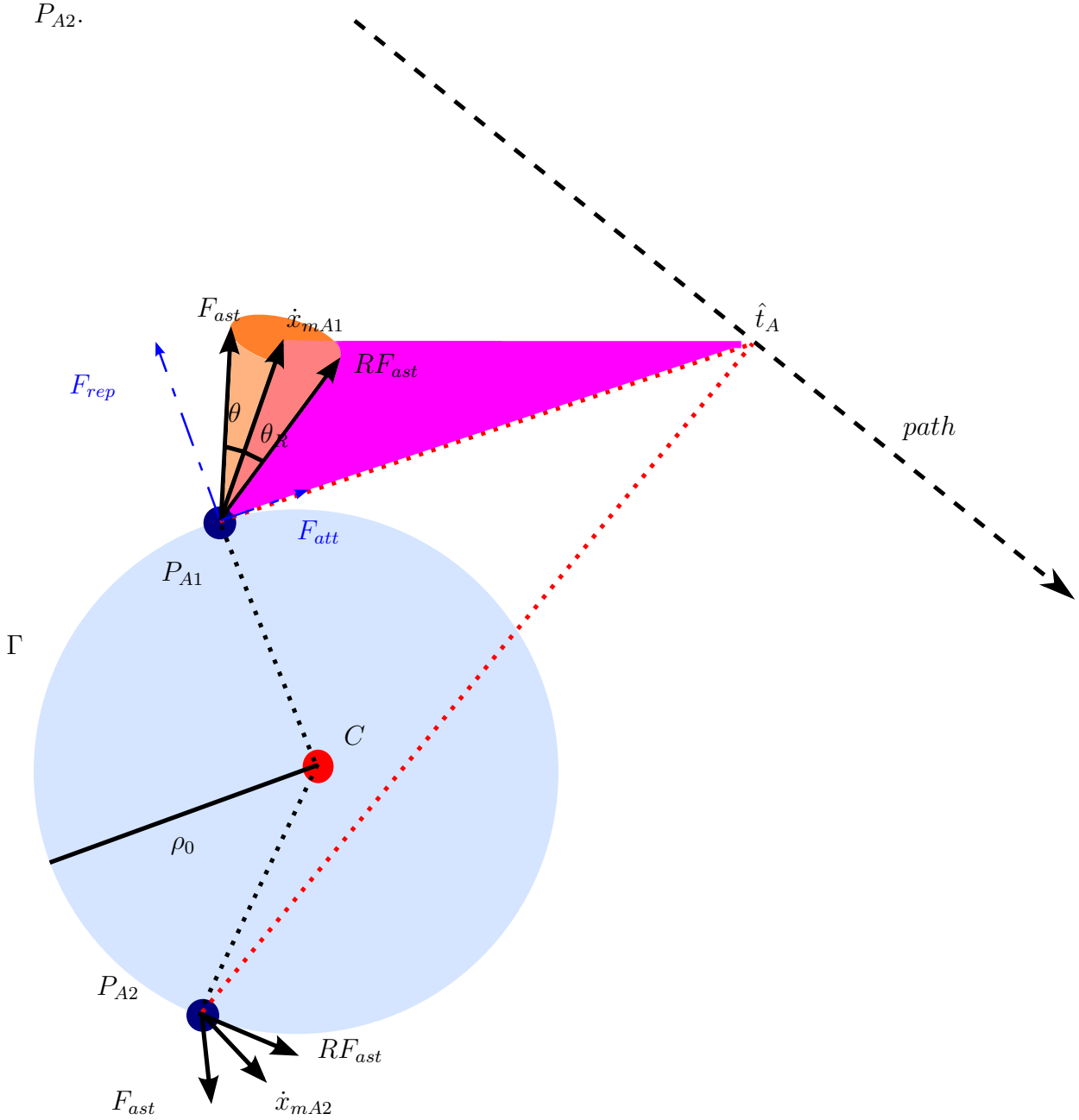


Figure 2.7: Example of optimization of the total assistive force direction.

According to these considerations, the rotation of the virtual force can be computed exploiting Algorithm 1.

Algorithm 1 Rotate assistive force

Require: $\hat{t}$, x_m, $\dot{x}_m$, C

1: calculate $F_{ast} = F_{att} + F_{rep}$
2: calculate the maximum admissible rotation angle θ given $\dot{x}_m$ and F_{ast}
3: calculate the distance d_{los} between C and the segment $\overline{x_m\hat{t}}$, the norm d_O of $\overline{Cx_m}$ and the norm d_C of $\overline{C\hat{t}}$
4: **if** $d_{los} < d_O$ and $d_O < d_C$ **then**
set $\hat{t}_R$ as the projection of $\hat{t}$ on the plane normal to $\overline{Cx_m}$ and containing x_m
5: **else**
set $\hat{t}_R$ as $\hat{t}$
6: **end if**
7: calculate the unit vector lying on the plane defined by $\dot{x}_m$ and $\overline{x_m\hat{t}_R}$, closest to latter segment and rotated by up to θ w.r.t. $\dot{x}_m$

2.6 Discussion

Surgical robotics is an attractive and challenging field of application of many research and engineering disciplines, including mechanical design, control theory, computer science and, of course, medical sciences. Specifically designed robots and related control methods have been developed by both private companies and research institutes. The safety of the patient is always the primary concern. Passivity-based control strategies allow to guarantee the stability of the system and the safety of the interaction between the robot and the environment (e.g. the body of the patient).

In this chapter, three specific situations are addressed: the automatic adaptation to changing tissue stiffness, the transition from autonomous to teleoperated mode and the teleoperation of a robotic system with multiple arms with the help of virtual fixtures. These situations replicate real life cases, when the surgeon adapts the stiffness of her/his arm to penetrate tissues of different consistency and when, due to an unexpected event, the surgeon has to take over the control of the surgical robot.

To address the first case, an admittance control strategy characterized by a time-varying stiffness, damping and inertia to assure passivity during interaction with the human body has been presented. By properly controlling the energy exchanged during the action, the system is guaranteed to be passive for any choice of the stiffness and inertia matrices. Therefore the robot exhibits stable behavior, both in free motion and in interaction with partially unknown environments (e.g. the body of the patient). The results obtained for the admittance controller are then adapted for implementing a variable impedance control.

For the second case, a safe, reliable and accurate procedure to handle the risky transition phase between autonomous mode and teleoperation of a surgical robot has been

proposed. The control architecture is based on a two-layered bilateral teleoperation architecture that guarantees stability and smooth variation of the critical variables. The freedom in the choice of control parameters and functions can be exploited to adapt the proposed solution to different conditions.

Finally, the third case is addressed by showing that it is possible to embed virtual fixtures in a bilateral teleoperation system for a dual arms robot without violating passivity. A passivity preserving strategy that allows to change the direction of the assistive force in order to maximize the efficiency of the teleoperation system has been proposed.

Chapter 3

Software architecture

This chapter presents the distributed architecture designed to fit the software requirements when automating a surgical task and the selected implementation of this architecture. The first sections of the chapter present the open-source framework used to implement the architecture and the development process behind the software implementation of the robotic control system. Then, the following section describes the implementation of the design model and its exploitation to build the I-SUR software architecture.

3.1 Introduction

The automation of a surgical action involves multiple aspects such as sensing, robot control and user interfaces just to name the most important ones. All these "elementary blocks" are spatially distributed and, possibly, implemented by using different tools and software programs. For these reasons the following framework requirements are necessary:

- An intrinsic distributed architecture where different tools running on different operative systems (OS) can work together
- The architecture should be a component-based architecture where each component can "wrap" a sensor, an algorithm, an actuator, etc.
- Real-time communication of critical data among components has to be guaranteed; e.g. the command from the controller to the robot, the force measurement from the sensing to the controller, critical alert coming from a situation awareness module, etc.
- Possibility to change easily and at run-time the communication topology. This is a critical point because the middleware should allow to change the connections among components whenever the autonomous procedure has to be stopped and the robot has to be teleoperated by the surgeon

- The behavior of the components should be described at an abstract level by using state machines and UML diagrams

The kind of architectures that better fits the design requirements is known in literature as *Component-Based Software Engineering (CBSE)* [55]. The basic idea is to decompose the whole system under study into blocks (i.e. hardware and/or software elementary subsystems) with clearly defined interfaces. Well designed interfaces allow:

- Successful integration of heterogeneous subsystems
- Less problematic maintenance of complex systems where many subsystems have to communicate each other and can be switched on/off according to the particular state of the system
- Distributed development of components
- Re-usability of certified or well tested components implementing critical functionality

From this point of view, well defined interfaces can be seen as a synonymous of trusted components, i.e. high-quality designed components with clear and unambiguous input-output interfaces.

The motivations for CBSE in industrial applications can be summarized by the following two engineering rules: "buy before build" and "reuse before buy". In the I-SUR project, the main goal driving through the choice of CBSE is the possibility to easily put together components designed by different partners with different tools just by having a common middleware where data and functionality can be unambiguously exchanged.

This means that, thanks to well defined interfaces, there is no need to design ad-hoc adapters to make components fit. The corner stone to reach this goal is to find a common model of architecture among the partners that permits to conform the design of each subsystems to the same criteria (e.g. common interfaces).

When the interfaces are properly set, different components with the same interfaces but different implementations can be easily interchanged: in this way any time an improvement within the component is implemented, the connections with the other components are not affected. The interface is a sort of wrapper hiding the implementation (if algorithms are the matter) or the physical device (e.g. two different force sensors can have the same interface) and showing only the input-output functionality.

In this way, a large and complex system composed by atomic components can be built in an unambiguous fashion. Such solution should be generic and flexible enough to allow static and also dynamic linking: the idea is to be able to change the topology at run time

without stopping the system whenever dangerous and unexpected situations occur and it is required to switch, for example, from autonomous mode to teleoperation. Moreover, the component-based approach is extremely useful to validate each component. Indeed, it is possible to check the performance and the requirement specification related to any atomic subsystem.

A middleware tailored for teleoperated/autonomous systems using open source software has been developed to design the software architecture and to satisfy the requirements described above. The middleware is based on Orocos (Open RObot COntrol Software) [56] components and allows to define a common protocol for exchanging data. The final goal was to make the integration of devices and algorithms as easy as possible.

This chapter will focus on:

- The definition of a requirements engineering approach to software design for complex cognitive robotic systems, capable of autonomous execution of surgical tasks.
- The integration of sensing, cognition and control capabilities into a modular software system, whose architectural properties allow to enhance reconfigurabiltiy and reusabiltiy of its main components. Particular care is given to the implementation of robot motion planning, control and supervision, which is the most critical part of the system.
- The application of open-source software frameworks, tools and libraries whose efficiency and reliability is well-proven by several robotics research projects, namely: *Real-Time Toolkit (RTT)* and *Kinematics and Dynamics Library (KDL)*, parts of the Orocos project, respectively used to implement and deploy software components and to implement robot control algorithms; *rFSM* [57], used to implement supervisory control based on hierarchical finite state machines; *Open Motion Planning Library (OMPL)* [58] and *Flexible Collision Library (FCL)* [59], used to implement online collision-free path planning for multiple robotic arms.

In Chapter 4 the experimental validation of the proposed software architecture will be presented. Even though the experiments are performed on a surgical robot prototype, the features and use cases of the software architecture are appealing even for robotics and automation applications in totally different domains.

3.1.1 Outline

The outline of the Chapter is as follows. Section 3.2 introduces the most important features of the Orocos framework while Section 3.3 describes the development process

that is behind the software architecture of the I-SUR project. Section 3.4 describes the requirements engineering approach being used and the formal verification of behavioral requirements. Finally, in Section 3.5 the proposed system architecture and its component-based software implementation are presented.

3.2 Orocos as a Middleware

Orocos is an open Source C++ software framework for building real-time component-based applications in automation and robotics. Orocos has been developed within a European project started on 2000 at K.U. Leuven, Belgium, in collaboration with LAAS Toulouse, France, and KTH, Sweden. It is a component-based architecture having the following advantages with respect to standard client-server socket communication:

- possibility to use real-time implementation of CORBA as communication protocol,
- possibility to change the communication topology among the different components without stopping the whole system (i.e. run-time reconfiguration),
- possibility to describe the behavior of the components by using state machines,
- multi-platform support.

In the following, the most important features and the "philosophy" behind Orocos will be briefly summarized. The interested reader can find more details and examples on Orocos in [56], to which the rest of this Section refers.

The core of Orocos is the *Toolchain*: the middleware connecting the applications (software components) and the operating systems through real-time (or not) communication (Fig. 3.1).

The system is decomposed into components (or *tasks* using the Orocos language) that are the basic units of functionality that, connected together, compose the application. Each component represents a part (hardware or software) of the system and interacts with the other elements through a *TaskContext* which is described by fundamental Orocos primitives (e.g. data ports, operations). The TaskContext defines the context in which an activity (a program) operates. It defines the interface of the component, its properties, its peer components and uses its *ExecutionEngine* to execute its programs and to process asynchronous messages. Figure 3.2 shows a schematic overview of a component expressed as a TaskContext. The component offers services through operations, and requests them through operation callers (methods). The Data Flow is the propagation of data from one task (or component) to another, where one producer can have multiple consumers and

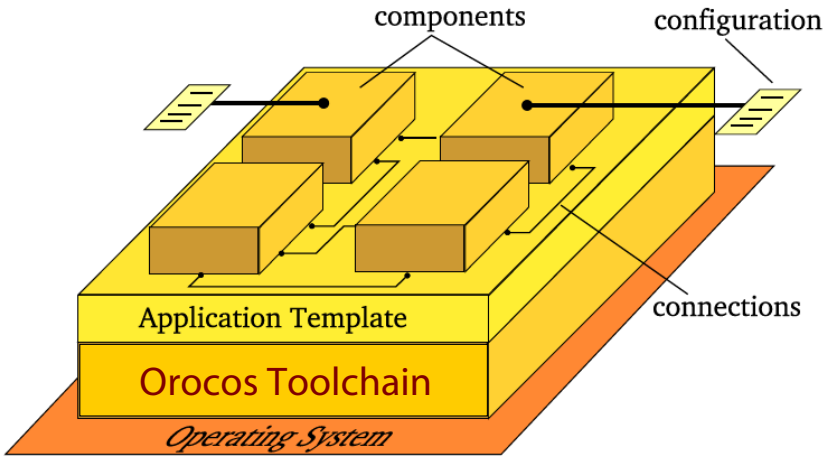


Figure 3.1: The Orocos Toolchain is referred to as *middleware* because it sits between the applications and the Operating System.

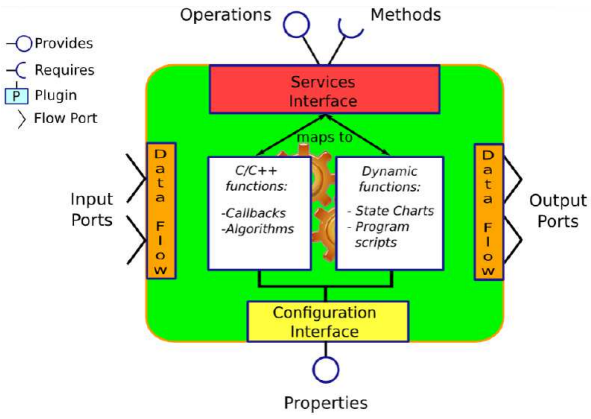


Figure 3.2: Schematic overview of a TaskContext.

the other way around. To interact with the TaskContext of different components, it is necessary to use a deployer (*DeploymentComponent*) that allows to explore, execute and debug components. Components could be created at run-time (dynamic deployment), even though in the present context, will be considered only the components created using C++ program (static deployment) and just executed by using the deployer. The component model in Orocos has been designed in such a way that it has the following characteristics:

- Lock free, thread-safe, inter-component communication in a single process
- Thread-safe, inter-process communication between (distributed) processes
- Communication between hard real-time and non real-time components
- Deterministic execution time during communication for the higher priority thread
- Synchronous and asynchronous communication between components
- Interfaces for run-time component introspection
- C++ class implementations and scripting interface for all the above points

Each component interacts with the others by its interface defined by

- operations and services,
- attributes and properties,
- data flow ports,
- peer components.

When a new component is created, it can be in many different states as reported in Fig. 3.3. The available states are

PreOperational: state of the component after the construction,

Stopped [S]: the component has been initialized by the script command `configure()`. This command is implemented in the C++ function `configureHook()`,

Running [R]: the component is running. The ExecutionEngine calls the function `updateHook()` in real-time or in a event-driven base,

Exception [E]: a run-time error occurs in some `*Hook()` functions

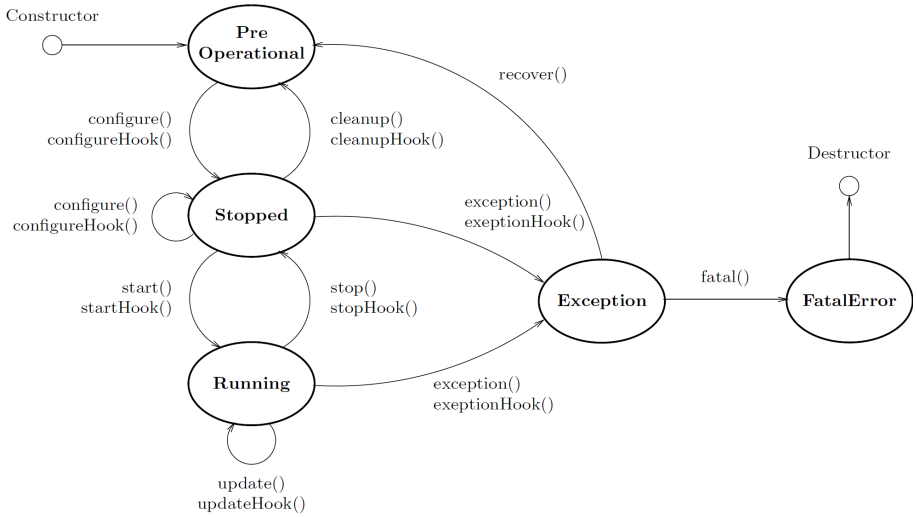


Figure 3.3: State diagram of the component (task)

FatalError [F]: the component enters this state when the ExecutionEngine is stopped. The component can only be deleted by the destructor

An example of code where the hooks are defined is reported in the following listing: the hooks are nothing else that C++ functions called by the ExecutionEngine when the component is in a certain state or when a transition occurs.

```
class myComponent
    : public TaskContext
  {
  public:
       myComponent(std::string name)
         : TaskContext(name)
       {
          // ...
       }

       /**
        * Configuration code.
        *       Return true for successful configuration.
        *       Return false to abort configuration.
        */
       bool configureHook() {
          // ...
          return true;
       }

       /**
        * Start up code.
        *       Return true for successful start up.
        *       Return false to abort start up.
        */
       bool startHook() {
          // ...
          return true;
       }

       /**
        * Code of the Execution Engine.
        */
       void updateHook() {
```

```
        // user stuff here ...
    }

    /**
     * Stop code.
     */
    void stopHook() {
        // ...
    }

    /**
     * Cleanup code.
     */
    void cleanupHook() {
        // ...
    }
};
```

The block diagram in Fig. 3.3 explains pictorially the state machine of any component and the possible transitions among states. The *Constructor* and *Destructor* are called when the component is created and destroyed, respectively, following the same rules in C++.

When the component is in the state **Running** the function `updateHook()` is called by the ExecutionEngine. According to how the component is defined, such function can be executed synchronously (i.e. at a fixed sample time) or asynchronously (i.e. event-based).

One of the crucial characteristic of any component is the definition of ***input and output ports*** used to interact with the other components. Any port is defined by three elements:

- a unique name (uniqueness within the component scope),
- the type of the exchanged data,
- input/output flow.

When the ports are defined, it is required to describe how components offer and ask services to the other components they are connected with. The ***services*** (i.e. methods or functions) are defined in the public section of the component and, through the command `addOperation`, made available to the other components. According to where the function is executed, the command `addOperation` can have the *OwnThread* argument, when the process is executed in the thread of the ExecutionEngine that offers the method, or the *ClientThread* argument, when the process is executed in the thread of the ExecutionEngine that calls the method.

Sometimes, it is also important to get or set the ***properties*** and the ***attributes*** of a component. They are defined in the component public interface using the keywords `addAttribute`, `addConstant` and `addProperty`. These variables are read and written by using the `set()` and `get()` methods. The properties can be read/written in the same way

as the attributes, even though there is also the possibility to read and write properties from or to XML files. This feature can be extremely useful when the data located in the file are, for example, the control parameters.

To ***connect components*** it is necessary to make each `TaskContext` communicates with the others. The `addPeer` (uni-directional connection) and `connectPeers` (bi-directional connection) functions do this job allowing to use other components interface. Using the command `connectPorts`, a connection among the ports with the same name of two components is established.

3.3 Development process

Validation-oriented design is mandatory for the application domain of surgical robotics. Therefore, design specifications for control algorithms and supervisory/coordination logic have been formalized using a requirements engineering approach, which is a more and more recommended practice for safety-critical systems design (see [35]). In particular, the applied methodology is developed as follows:

1. ***Requirements collection***. A group of expert surgeons is interviewed on the objectives of the surgical process, the main procedures ("best practice") to be performed, the elements of the domain and the critical events related to the surgical actions.

2. ***Requirements engineering***. Surgical requirements are expressed using a goal-oriented methodology called FLAGS (*Fuzzy Live Adaptive Goals for Self-adaptive systems*, see [60]), that has two main features: it is focused on real objectives of an operation and on complications that may arise during its execution; it is based on a formal language. Indeed, the *goal model* is a set of formal properties expressed in the *Alloy language* (see [61]), a specification language for expressing complex structural and behavioral constraints for a software system, based on *First-Order Logic* (FOL) and *Linear Temporal Logic* (LTL, [62]). For example, a leaf goal of the cryoablation procedure requires to avoid *forbidden regions* (i.e. bones, nerves, other organs) during needle insertion, which can be specified as follows:

   $$MP => !(FR \ \wedge \ (FR.needle = MP.needle))$$

   This formula asserts that every time a movement is performed (event MP), the *needle* entity associated to the movement must not touch a forbidden regions (event FR).

3. ***Operationalization***. The goal model is transformed into a sequence of operations and adaptations, satisfying the goals of the surgical procedure. This task is formally

defined as a constraint satisfaction problem, whose solution is obtained by using the SAT-solver Kodkod embedded in the Alloy Analyzer tool, as shown in [63]. As a result, the tool provides a sequential model equivalent to the traces of a state machine, representing the whole system behavior that guarantees the achievement of the root goal.

4. ***Modular System Design***. The state model obtained after the goal-oriented analysis is refined and partitioned into the structural units of the overall automated system, implementing a collaborative and coordinated behavior compatible with the requirements. This task is performed applying decomposition methods from classical discrete systems theory and using UML (*Unified Modeling Language*, [64]) as a modeling tool, to ease software-oriented design specification.

5. ***System Verification***. Formal tools like Model Checking (see [65] and the related SMV tool [66]) are applied to verify that the UML system model preserves the properties expressed by the goal model. This task requires the formalization of an appropriate semantics of the UML behavioral specification (i.e. State Diagrams of system components), since the UML language itself does not include a strictly formal one.

6. ***Experimental Validation***. This task is of course mandatory before the clinical use of automated devices. Though the aim of the proposed research is to evaluate the feasibility of surgical robotics automation only using artificial phantoms, demonstrations involving expert surgeons directly interacting with the system are primarily important even to achieve this goal.

3.4 System design and model checking

The autonomous system being designed in the I-SUR project is supervised and controlled by the following modules, corresponding also to software units deployed on different computational platforms:

- ***Surgical Interface***

  Software-intensive system allowing humans (i.e. surgeons and technicians) to drive the system during both the pre-operative and the intra-operative phases. In the first one, the focus is on detailed planning of the surgical intervention (e.g. for the puncturing task, enumeration and placement of cryoablation needles for maximal tumor coverage thanks to the *Cryo-planner* that will be described in Section 4.3.1).

During the execution of operations, the interface should provide real-time visual navigation of the surgical scenario and, if necessary, let the surgeons take control of the system (e.g. by switching to teleoperated mode).

- ***Robot Controllers***

  Units implementing control of surgical actions and tasks sequencing during the intra-operative phase. The event-driven behavior extracted from the goal model is mapped into the control logic of each robot, specified by a UML State Diagram. Safety-critical requirements put a strong demand for strict coordination of these components with both the Surgical Interface and the Sensing/Reasoning module.

- ***Sensing System and Reasoning for Situation Awareness***

  Composite sub-system implementing advanced algorithms to provide support to the planning task, during the pre-operative phase, and prompt identification of anatomical changes or discrepancy between the tasks being executed and the *nominal* surgical plan. Bayesian Networks and Particle Filters (see [67]) are used to detect the occurrence of undesired events and critical situations, so that appropriate corrective actions can be triggered.

The interaction among such system components has been specified with the help of UML Sequence Diagrams, which represent scenarios compatible with a given collaborative behavioral specification. As an example, Fig. 3.4 shows an admissible scenario for the cryoablation execution, focused on needle insertion under US-based monitoring. The scenario specifies the initial setup of the surgical task, in which pre-operative medical imaging data are processed by the cryoablation planning algorithm that will be presented in Section 4.3.1, whose result is the optimal placement of cryoprobe needles to obtain full tumor coverage with the expected iceball, without interference with other organs (i.e. *forbidden regions*). The needles placement is referred to the center of the tumor, therefore the task plan, once validated by the surgeon, must be adapted to the operative scenario by means of the registered coordinate transformations calculated by the Sensing/Reasoning module. During the needle insertion task, the US probe is first placed on the surface of the body, aligned with the expected needle tip trajectory, and then the planned needles are mounted on the robot end-effector and automatically inserted.

The complete behavioral specification of the robot control and supervision units is given by UML State Diagrams associated to the control logic for the robot holding the needle and for the robot holding the US probe. Figure 3.5 shows the hierarchical state machine related to the needle inserting robot.

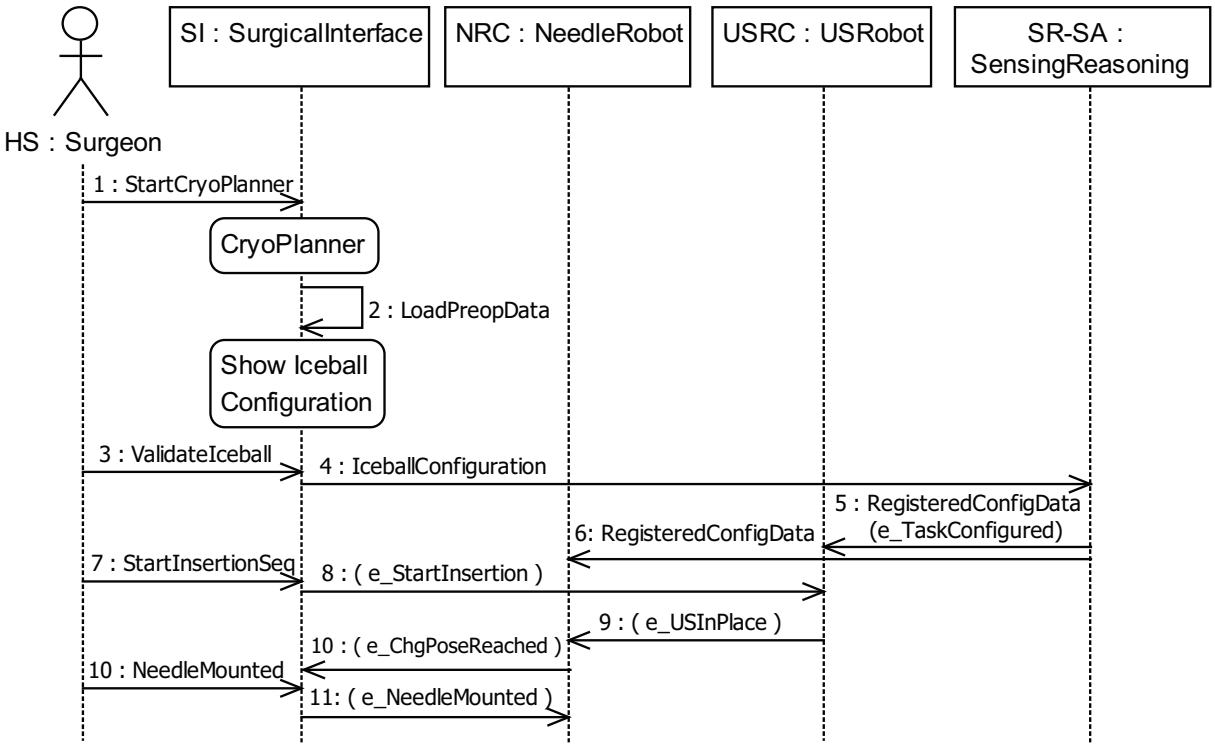


Figure 3.4: UML Sequence Diagram of the interaction among system components during setup of cryoprobe needle insertion.

As can be seen, the hierarchical features of UML State Diagrams allow to embed exception handling mechanisms, by means of transitions exiting composite states. In both state machines, in fact, the robotic task can be stopped because of an exception event, that can be triggered either by the surgeons, through the Surgical Interface, or by the Sensing/Reasoning and Situation Awareness module. In particular, the latter is in charge of detecting if the needle is too close or even touching a forbidded region, like for example a bone, a nerve or another organ not involved in the cryoablation, by means of real-time monitoring of the needle motion within an anatomical atlas of the patient. Another undesired event is triggered if any force value measured by the sensors exceeds a given limit. Whatever is the exception event, if the task execution can be restarted after appropriate validation of the surgeons, the transitions marked by the `e_taskRecovered` event are executed. Eventually, the system allows the surgeon to switch to a teleoperated mode. Control algorithms preserving stability during such a mode switching have been specifically designed and described in Section 2.4.

Similar diagrams have been designed to address the suturing task.

Formal verification of the UML design model requires the definition, first of all, of its operational semantics, according to the execution model of the target implementation framework. In particular, since the proposed UML design has been implemented using the component-based Orocos framework [56] and rFSM, an execution engine for hierarchical state machines written in Lua language, the peculiar features of the latter must be carefully considered. A detailed analysis of the semantics of the rFSM model and its differences with the one informally described by the UML standard can be found in [57]. Summarizing the main points of the reference:

- in UML events are supposed to be stored in a queue and processed one at a time to evaluate the enabled transition set of the state machine. Instead, rFSM collects all events occurred since the last executed machine *step* and uses them to evaluate enabled transitions and execute the next step, after which the whole set of occurred events is cleared;

- conflicting transitions are solved according to different structural priority schemes: UML gives higher priority to transitions whose source state is at lower hierarchical levels, while rFSM reverts the rule;

- rFSM does not support concurrency (i.e. so-called AND-states), assuming that this feature is managed by the component-based execution framework of Orocos.

Assuming that an rFSM machine is embedded into a given Orocos component with input and output *event ports* to interact with rFSM machines in other components, it is

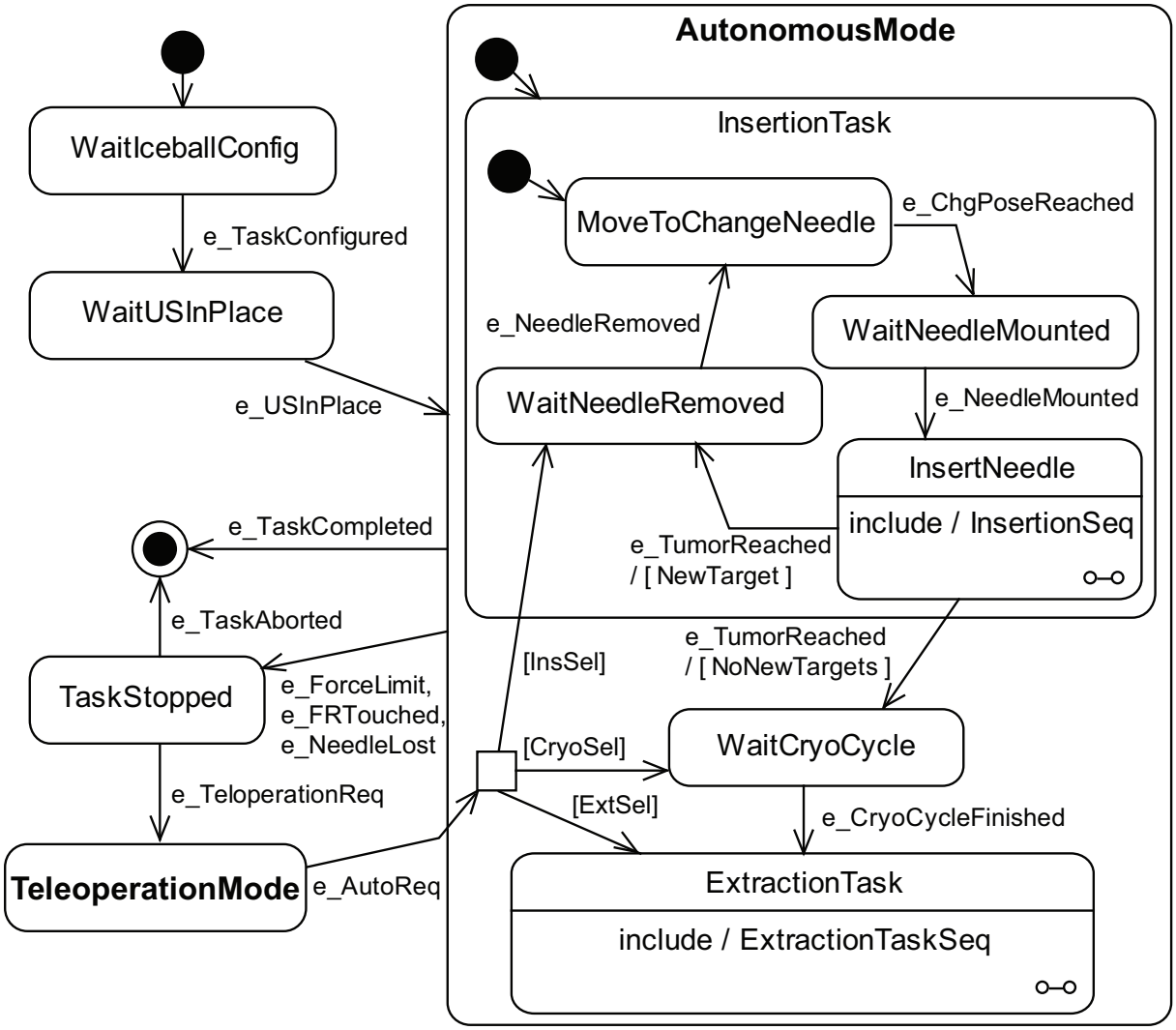


Figure 3.5: UML State Diagram of the behavioral specification for the controller of the robot holding the needle.

possible to formalize the UML design model implemented in Orocos-rFSM as a modular transition system, following the approach described in [68]. In particular, the formal model of a component embedding an rFSM machine is a structure:

$$C = (S, T, P, r) \tag{3.1}$$

where S is a (hierarchical) set of *states*, T is a set of *transitions*, $P = P^I \cup P^O$ is a set of "port" variables, each one of a given data type (including *event*), and $r \in S$ is the root state. The full system is then defined as an ordered set of components and interconnections, together with a *scheduling function*. Such a formal model can be easily translated into the input language of the Cadence SMV (Symbolic Model Verifier) tool [66], which is still one of the model checkers that handles most efficiently the well-known state-space explosion problem. The SMV language allows to describe a finite state system with constructs to declare modules and data-types, supports both boolean and integer arithmetic and has specific constructs to initialize state variables and to assign them the next value in a computational path. Moreover, SMV assumes a synchronous model of execution and allows users to specify desired properties using either Computation Tree Logic (CTL) or Linear Temporal Logic (LTL).

It is important to remark that the previously described rFSM events collection mechanism is quite different from the PLC-like execution model described in [68] and requires a specific adaptation. In particular, each event must be associated to an SMV module, whose internal boolean state is set true if the event has occurred and is reset when the event is cleared by the execution of the step of its "container" rFSM module. The module in SMV code is the following:

```
MODULE rFSM_EV(Event, Clear)
VAR
  Occurred : boolean;
ASSIGN
init(Occurred):=0;
next(Occurred):= case
                 !Occurred & Event : 1;
                 Occurred & Clear : 0;
                 1 : Occurred;
                 esac;
```

The SMV module related to an rFSM machine will include an `rFSM_EV` for each input and output event:

```
MODULE rFSM_Robot(ExecStep,e_STOP,..)
VAR
e_STOP_ev: rFSM_EV(e_STOP, (Exec = FINISHED));
..
Exec : {IDLE, STEP, FINISHED};
```

The module has also a boolean input `ExecStep` that triggers the execution of its step. This input will be set by previously mentioned scheduling function of the composite system. Input events are cleared when the step execution is completed, a condition defined by a value of the enumerated variable `Exec`.

The UML State Diagram specifying the behavior of an rFSM module is encoded preserving the hierarchy of states into variables with enumerated values like:

```
Root : {InitRobot, NeedleTaskAuto, TaskStopped, ..};
SUBNeedleTaskAuto : {Ready, ..., MotionPaused};
```

and the current configuration of the machine and the set of enabled and firable (i.e. higher priority) transitions is evaluated with predicates defined as follows:

```
INInitRobot := (Root = InitRobot);
INReady := INNeedleTaskAuto &
           (SUBNeedleTaskAuto = Ready);
ENTrans1 := INStateXX & Trigger & Guard;
CONFLTrans1 := ENTrans2 |  ENTrans3 | ..;
FIRABLETrans1  := ENTrans1 & !CONFLTrans1;
```

Finally, initialization and execution of a step can be translated as follows:

```
init(Root) := InitRobot;  -- default state
init(SUBState1) := Ready; -- default state
default{
      next(Root) := Root; -- no change
      next(SUBNeedleTaskAuto) := SUBNeedleTaskAuto;
...}
in case{
     (Exec=STEP) & FIRABLETrans1 : {
         next(Root):= NeedleTaskAuto;
           ...}
     (Exec=STEP) & FIRABLETrans2 : {
           ...}
...}
```

An SMV program is completed by the declaration of a `main` module (i.e. the top-level) and by the specification of desired properties of the system. As said before, the desired properties can be expressed using LTL, in the same way that leaf goals are specified by the goal model of the requirements specification.

Of course, the design model must be not only verified, but also implemented and validated by surgeons, on the basis of experiments on phantoms or virtual environments and subsequent evaluation of the task execution. Next sections describe a component-based software implementation of the proposed system, highlighting the benefits of its reconfigurability properties on some interesting use case scenarios and experiments.

3.5 System architecture

The implementation of the design model presented in the previous section is organized according to the classification of system components shown in Figure 3.6. Such a classification reflects the definition of the three main modules previously described and further separates software components into those mainly event-driven, acting at *coordination* level, and those performing data-driven *computations*, following a separation of concerns as suggested in [69]. In particular, the Reasoning module is actually composed by *Sensing* software parts, implementing for example real-time image processing algorithms to extract interesting features (i.e. needle and tumor for the cryoablation task or wound edges for the suturing task) from US images or camera images, and *Situation Awareness* components, implementing Bayesian Networks processing data received from Sensing and *Robot Control* components and detecting undesired events and exceptions. The event-driven behavior specified by UML State Diagrams (see e.g. Figure 3.5) is implemented by *Task Supervisor* components, coordinating the operations of the robots and the overall task execution. Software modules implementing motion planning and control algorithms, which are mainly data-driven and run as real-time periodic tasks, are classified as *Robot Control* components. Finally, the *Surgical Interface* contains all the software parts related to the interaction with human operators.

The rest of this section is focused on components for robot motion planning, control and supervision, a critical sub-system in which the benefits of the proposed software framework in terms of distribution, reconfigurability and coordination are more useful.

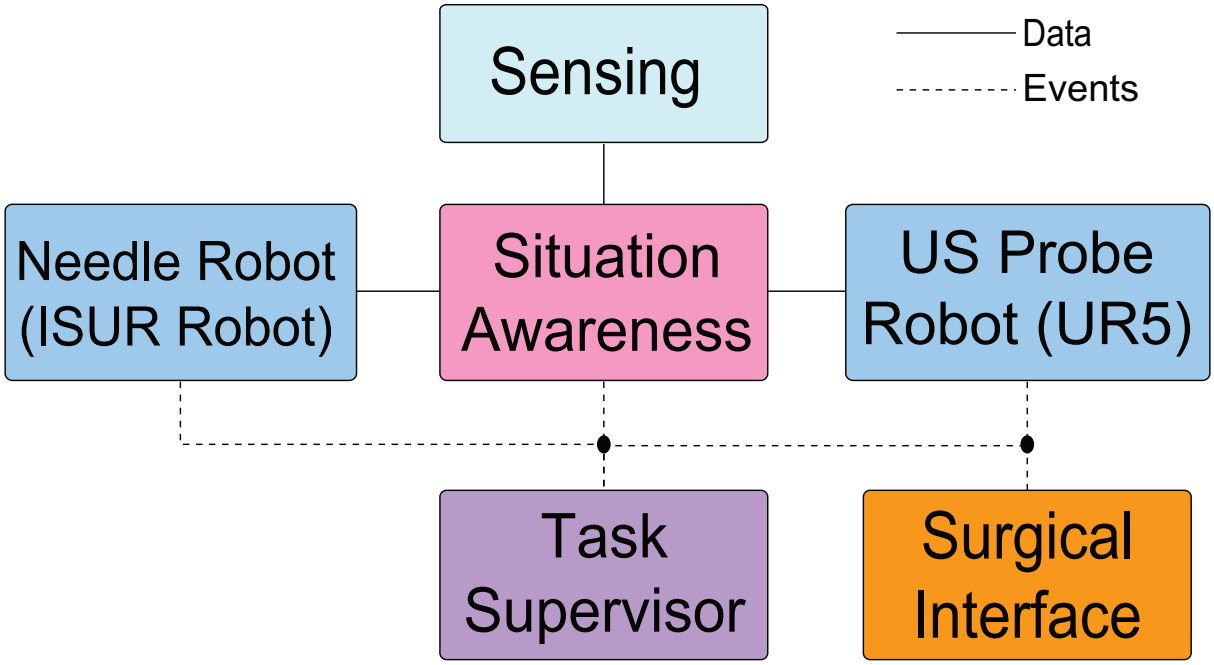
The planning and control module is in charge of:

1. searching *valid* Cartesian paths for robot motion, satisfying task requirements (i.e. goal poses for needle insertion reaching the tumoral mass for the puncturing task or stitches points for the suturing task) and avoiding collisions between the two robots, between the two end-effectors and with other objects inside the workspace;

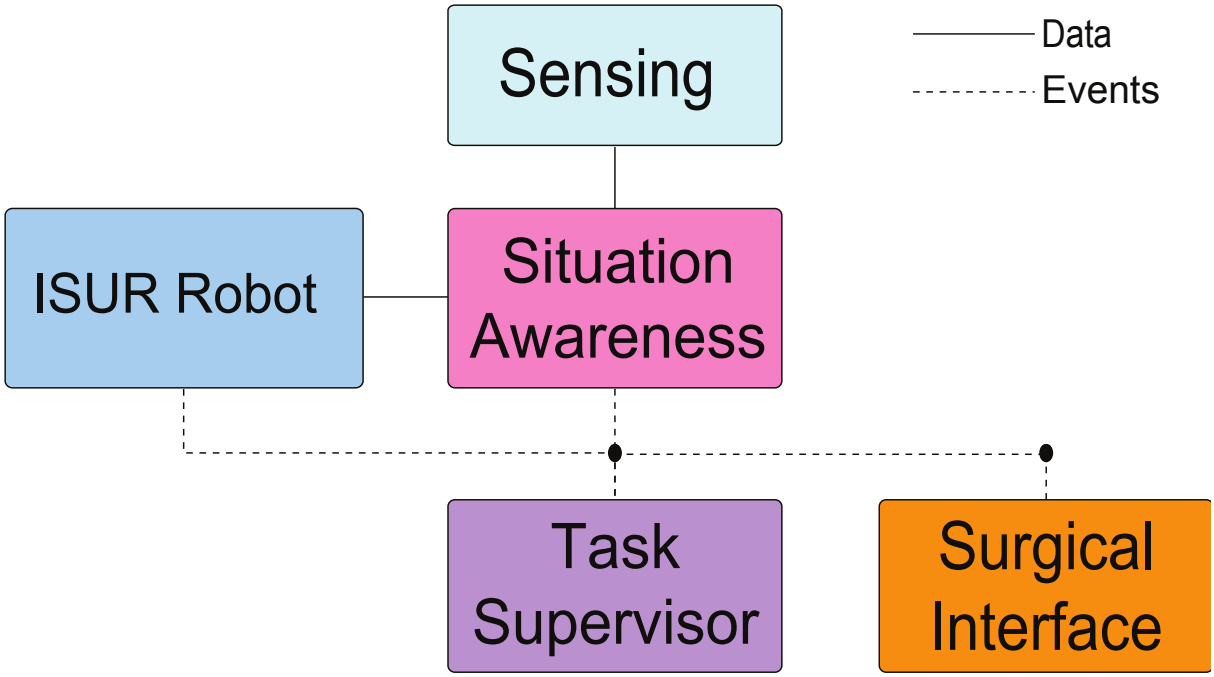
2. generating timed trajectories satisfying dynamic constrains (i.e. maximum velocity and acceleration);

3. generating low-level commands for the control hardware, in order to track the desired trajectory.

Supervision and coordination of the two robots or of the two end-effector of the I-SUR robot is instead performed by components implementing the behavioral task specification described in Section 3.4. Finally, the configuration of tasks, which is composed of



(a) Puncturing task



(b) Suturing task

Figure 3.6: General scheme of the proposed Software Architecture.

state machines, control parameters, pre-operative data and intra-operative adaptations is managed at deployment level, by means of specifically defined scripts.

The features required to implement such architecture are substantially summarized by the following software primitives: *components* (with *operations* and *properties*), *data flows*, *events* and *finite state machines*. Such features, eventually called with different names, are supported by several software frameworks focused on robotic applications. In addition to Orocos that was previously introduced, ROS [70], YARP [71], OpenRTM [72], Orca [73] and GenoM [74] could be mentioned. Among others, ROS seems to be the one with the largest user base, the largest list of existing modules, and the easiest in terms of integration among different software and hardware components. However, it does not fulfill real-time requirements, which is instead a central feature of Orocos. Even if Orocos components and ROS nodes can easily interoperate, the proposed architecture is strictly based on the former ones. In this way, hard real-time behavior is guaranteed and, on the other hand, it is still possible to extend the system using ROS nodes for those parts in which real-time is not mandatory (e.g. sensing for intra-operative registration).

As described in [75], an Orocos component is a basic unit of functionality which executes one or more (real-time) programs in a single thread. Orocos allows to implement a component as a `TaskContext` C++ class, whose interaction with other components is defined by means of three primitives: Properties, Operations and Data Ports. The Orocos component model enables lock free, thread-safe communications between (distributed) components and deterministic real-time execution, provided that the underlying operating system supports adequate scheduling features. Orocos components can be distributed over a network, thanks to the support of CORBA [76] interoperability. Moreover, the `ExecutionEngine` associated to each `TaskContext` and the Orocos deployment scripting interface provide flexible and powerful tools for setup and run-time reconfiguration of components.

In the following, the components developed for real-time robot control are introduced. Since the main objective of the proposed architecture is to embed autonomous behavior into a surgical robotics setup, the control system has a *nominal* mode of operation in which a complete surgical task is automatically executed, according to a planned sequence, by loading target poses for the tools mounted on end-effector of the robots, computing collision-free paths reaching those targets and generating and tracking the planned trajectories with robot motions. However, undesired events and direct surgeon requests may force the system switching to a *teleoperated* mode, in which the motion planning subsystem is not active and the surgeon takes control of the robots by means of dedicated haptic interfaces. Force feedback to such haptic devices is provided implementing the

bilateral teleoperation control algorithms presented in Section 2.4. It is also important to remark that the interconnection of control software components is reconfigured during mode switching, as shown by Fig. 3.7 and Fig. 3.8. Components that are active in both modes are depicted with solid borders, while those activated only in a given operating mode have dashed borders.

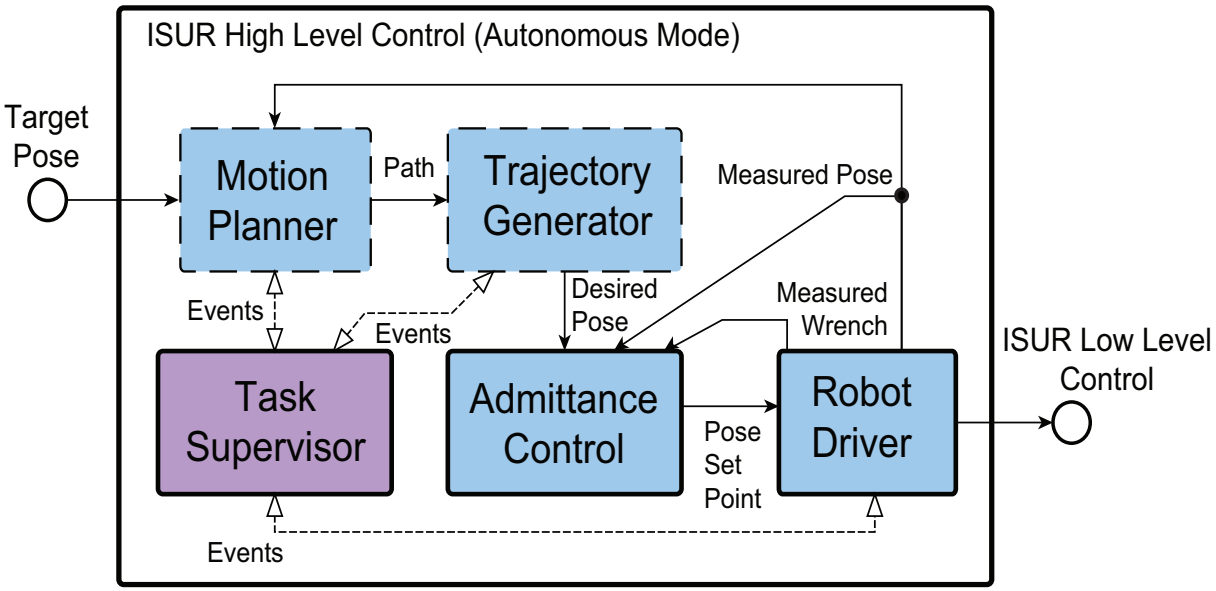


Figure 3.7: Interconnection scheme of robot motion panning and control components in autonomous mode.

As can be seen, the Admittance Control module, whose role is to implement the desired operational space behavior described in Section 2.3.1, is designed to accept either pose or wrench (i.e. 6 DOF force/torque vector) commands, according to the operating mode of the system. The rest of this section describes more precisely the features of the developed software components.

- ***Task Supervisor***

  The desired task is specified by a state machine, as described in Section 3.4 with the example of Fig. 3.5. This state machine is translated in Lua using the rFSM framework, then Lua files are loaded by an Orocos component, instantiated from the class `OCL::LuaComponent` of the RTT-Lua library [77], that acts therefore as a supervisor of the control architecture. Any other component of the system is expected to generate specific events, such as the completion of a motion step or inputs from the Surgical Interface, that are processed by the supervisor to detect the progress of the task and to coordinate the overall behavior. In this way, each software component is

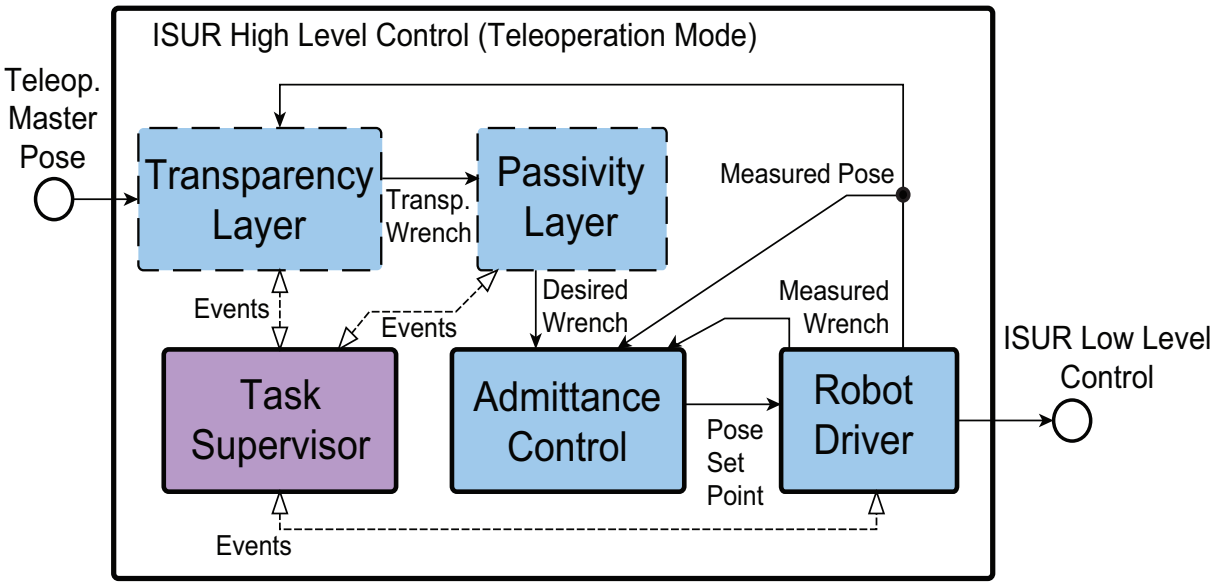


Figure 3.8: Interconnection scheme of robot motion panning and control components in teleoperated mode.

developed to provide context-independent basic functionalities and its reusability is increased, since the coordination and configuration is demanded to the Task Supervisor. In fact, the Task Supervisor has complete knowledge of all the properties and the operations provided by the components of the architecture, so that the system can be reconfigured online in accordance with the state of the task. This task-dependent reconfiguration is directly managed by the supervisor component thanks to the rFSM execution engine and the Orocos RTT-Lua extension library, which provides also the software functions for components deployment and ports connection. According to the definitions included in [69], the proposed Task Supervisor plays the roles of `Configurator`, `Coordinator` and `Composer` entities. Even though, as discussed in the reference, the 5C separation of concerns principle suggests that such entities should be implemented as different components, the centralized approach adopted here reduces the need for event communications and simplifies the development. The complexity issue related to large task specification/configuration programs has been addressed by splitting them into multiple Lua files, many of which are automatically generated.

- ***Task Frames Generator***

  This component contains all the computations required to transform the data generated by pre-operative planning tools, using transformation matrices calculated during both offline and online registration (i.e. transformation between the refer-

ence frames of the two robots, transformation from the end-effector of each robot to the specific tool tip, transformation from US image coordinates to the robot frame, transformation of the origin of patient/phantom 3D model to the robot frame), so that the proper target points are assigned to the motion planner in each state of the task. For example, in the case of the *Cryo-planner*, needles placement is referred to the center of the tumor, therefore the task plan, once validated by the surgeon, must be adapted to the operative scenario by means of the registered frames transformations calculated by the Sensing module. It is important to note that this component is the only one, together with the Task Supervisor, that contains an intrinsic knowledge of the whole task setup. On the other hand, the Task Frames Generator is a pure `Functional Entity`, like other components described next, according to the 5C separation of concerns.

- ***Motion Planner***

  This component is in charge of planning valid Cartesian paths for a robot. In the puncturing task, the complete system includes two instances of the Motion Planner component, while in the suturing task the system includes only one Motion Planner component that computes the path for the whole dual-arm system. The path is calculated taking into account that each pose of the path must be reachable by the robot (i.e. within its workspace) and that the tool tip must not collide with anything in the operational space of the robot, including its own links (*self-collisions*), those of the other robot or of the other end-effector (*inter-robot collisions*), and other objects (*obstacle collisions*).

  The software implementation makes use of the Open Motion Planning Library (OMPL) [58] and, in particular, of the *RRT-connect* [78] sampling-based algorithm. For the puncturing task, where two independent paths are required (one for each robot), the algorithm grows two RRTs (Rapidly-exploring Random Tree), from the start and the goal poses, by sampling random states of the path space, which is in this case the *special Euclidean group* SE(3), validates them using inverse kinematics algorithms based on Orocos KDL [79] and collision-checking algorithms of the *Flexible Collision Library (FCL)* [59], then tries to find the path solution connecting the two RRTs. For the suturing task, two end-effectors are mounted on the same structure. Thus, the two arms are constrained each other and the Motion Planner has to compute only one path in the state space SE(3)xSE(3) (left/right arm tips). As for the puncturing, state samples are validated by calculating a feasible inverse kinematics solution (i.e. within joint limits) and checking for collisions (among robot links and between robot and environment) using the random sampling algorithm

RRT-Connect in OMPL.

The full set of possibly colliding objects can be defined by a list of 3D CAD models or by text files describing basic geometric shapes (e.g. spheres, cones or boxes). Moreover, each object of the set is associated to an enable flag, which can be modified online in order to vary the constraints (i.e. the set of objects actually considered as obstacles to avoid) and then the behavior of the robot with respect to the environment. Finally, even the minimum allowed distance between the robot and a specific type of object is a parameter that can be changed by the Task Supervisor in accordance with the task state.

- ***Trajectory Generator***

  This component interacts with the Motion Planner and the Task Supervisor in three steps. In the first one, the component waits for a path described by two (i.e. start and goal of a linear path) or more (for more complex paths) poses and as soon as a path is received it is stored in an internal buffer. Then, if a `genTraj` operation is called, the Trajectory Generator calculates the actual trajectory given the geometric path and velocity and acceleration constraints, defined as properties of the component. Finally, once that the trajectory is prepared the component waits for a request by the Task Supervisor, by means of the `startMove` operation, to start the data flow output of Cartesian set-points with a fixed sample rate, which is also configurable as property. Once that the trajectory is completed an event `e_MoveFinished` is generated and the component goes back to the first state, waiting for a new path.

- ***Transparency/Passivity Layer***

  An instance of each of these two components is activated by the Task Supervisor when a robot is switched from autonomous to teleoperated mode, in case an undesired event is detected by either the Sensing/Reasoning module or the surgeon. The name of the two components refers to the *Two-Layer* approach to bilateral teleoperation described in Section 2.4. The layered control system is designed to preserve the passivity property with respect to the autonomous/teleoperated mode switching. One instance of both the Transparency and the Passivity components is switched on for the master device, while one instance of the same components is switched on for the slave device. Master and slave Transparency components exchange position and velocity information for computing the desired coupling forces to be applied as defined in (2.49). These forces are then sent to the Passivity components whose role is to passively implement them using the energy stored in the

tanks, as described in Section 2.4.

Transparency and Passivity Layers are running in the architecture in alternation with the Motion Planner and the Trajectory Generator: the Supervisor is in charge of the transition from one to the other in accordance with the state of the task.

- ***Variable Admittance Control***

  Following the approach described in Section 2.3.1, a variable admittance controller has been implemented for the high-level control of the robot, introducing the possibility to modify online the stiffness of the interaction model without loosing passivity and, hence, stability of the closed-loop system. Thanks to this dynamic behavior, the controller is adapted during the execution of the task, so that the robot is, for example, more compliant when approaching the needle to the skin and stiffer when the needle is pushed towards the final target. The parameters of the controller are changed directly by the Task Supervisor.

- ***Robot Driver***

  Each robot is associated to a dedicated driver component, hiding hardware-dependent details related to the data structures and communication protocols required to receive position/force measurements and send low-level motion commands. Newer components developed in the future will be easily integrated in the system, provided that they comply with the specified interface.

3.6 Discussion

The achievement of complete automation of surgical robots requires advanced developments in different technological sectors, especially those related to information science. Indeed, an autonomous robotic system may help human surgeons only combining advanced sensing, cognition and control capabilities, developed according to rigorous assessment of surgical requirements, formal specification of robotic system behavior and software design and implementation based on solid tools and frameworks.

An autonomous or semi-autonomous surgical environment is a really complex system. Several devices and different subsystems have to be put together harmoniously and following specific requirements coming from the knowledge of the surgeons. The possibility of reusing some parts of an existing validated system can really speed up the development of new systems and help technicians and scientists to start from a certain point and not from the beginning.

In this chapter, a robot control and coordination software architecture for the automation of simple surgical tasks is presented. Design specifications were defined using a requirements engineering approach, allowing a formal verification of behavioral requirements and the extraction of hierarchical finite state machines describing robotic tasks. Then, the proposed architecture has been implemented using component-based design tools, namely Orocos RTT and rFSM, in order to properly handle properly the distributed nature of the system and apply state-of-the-art robotics software design principles.

Chapter 4

Case studies and experimental validation

This chapter describes the implementation of the control strategies and the software architecture presented in the previous chapters to address two particular case studies: the puncturing task and in particular the cryoablation of kidney tumors and the suturing of a planar wound. At the beginning of the chapter, a description of the robotics platform designed within the I-SUR project will be provided. Then, the chapter addresses the full case study of the puncturing task and the results of implementing the variable admittance/impedance control and the safe switch from autonomous to teleoperated mode. Finally a proof of concept of reusability of the whole architecture will be provided by showing the possible adaptation to address the suturing task and the results of implementing the teleoperation architecture for the dual arms robotic system will be discussed.

4.1 Introduction

Experimental tests were performed on the semi-autonomous robotic surgical system specifically developed within the I-SUR project in order to validate the theoretical findings presented in this thesis and show the possibility to autonomously perform basic surgical tasks.

In particular, a specific application of the needle insertion task, namely the percutaneous cryoablation of small kidney tumors, will be addressed more intensively, in order to emphasize the potential benefits of the proposed technology and the pre- and intra-operative analysis that it allows. At the same time, the surgical task should be considered as one possible case study, since modularity and reconfigurability of the proposed control software architecture allow to easily update the system, provided that the mechanical setup is updated accordingly, for the execution of other surgical tasks. As a proof of concept, a possible adaptation of the setup and of the software architecture to address a different surgical task, namely the suturing of a planar wound, will be shown in the final

section of this chapter. Since the suturing task needs to be performed using a dual arms system, implementation of the teleoperation system for a dual arms robotic system will be shown.

Percutaneous cryoablation requires the use of pre- and intra-operative images, acquired by CT, MRI or US to insert, through the skin, one or more cryoprobe needles into the tumoral mass to be destroyed and to check the real-time position of the needle tips inside the patient. Once the needles are correctly inserted, the cryoablation operation involves cycles of freezing and thaw to create an iceball covering and killing the tumoral cells, while preserving the healty tissue and the surrounding abdominal structures.

Thanks to the complexity of the task from an engineering point of view, percutaneous cryoablation represents a full case study to show and validate all the theoretical results presented in the previous chapter. In particular,

- ***Planning of the surgical task.*** The first step in automating a particular surgical task, is the definition of the target zone to be reached and the identification of anatomical features of the zone, of the surroundings and of the forbidden regions that must be avoided (ribs, nerves, vessels, surrounding organs). For example, the cryoablation procedure requires to select the correct number and placement of the cryoprobes in order to completey freeze the tumor, while minimizing the damage to the surrounding healthy tissue and satisfying all the possible constraints to needle insertion. To this aim, in the following the development of a computerized planning tool that automatically select the number and the correct location for cryoprobes insertion will be described.

- ***Software architecture.*** Developing and experimenting a full case study for a surgical task requires efforts from different research teams and, therefore, the integration of possibly different technologies. A distributed software architecture is required and has been developed on the basis of the results described in Chapter 3. The architecture behind the implementation of the puncturing task can be easily adapted to perform other surgical tasks (e.g. the suturing of a planar wound).

- ***Interaction control to execute the task.*** When executing the surgical task, the robot has to interact with a mostly unknown and usually deformable or non homogeneous environment. The admittance and impedance control strategies with time-varying admittance/impedance parameters presented in Section 2.3 are exploited to ensure a stable and safe interaction between the robot and the human body when performing the task. Furthermore, due to an unexpected event, the surgeon may need to take the control of the robot. Thus, the bilateral control

architecture described in Section 2.4 has been implemented to ensure a safe behavior during the transition between autonomy and teleoperation and still retain high performance control after the switch.

4.1.1 Outline

The outline of the Chapter is as follows. Section 4.2 provides a description of the I-SUR robot. In Section 4.3 the results of automating the execution of the puncturing task are described to validate all the theoretical findings presented in this thesis. Then, Section 4.4 describes a second surgical task (i.e. the suturing of a planar wound), that provides a proof of concept of reusability and reconfigurability of the whole control architecture. In the same section, results of implementing the teleoperation architecture for a dual arms robotic system with virtual fixtures are shown.

4.2 The I-SUR robotic platform

Most of the existing robotic surgical platforms dedicated to autonomously achieving parts of or complete surgical procedures have been designed for one specific task, e.g. for joint replacement surgery (ROBODOC [5]) or prostate resection (Probot [4]). The automation of multiple surgical tasks with a single surgical platform is a new major challenge, which motivated the design of a versatile and dexterous robotic platform within the I-SUR project.

The I-SUR robotic platform was designed with the aim of autonomously performing relatively simple surgical actions, such as puncturing and suturing. The automation of the insertion of needles through the skin as required, for example, by a kidney tumor cryoablation, requires a large workspace to properly position and orient the needle over the tumoral area, a stiff structure to guide the needle during the insertion and the ability to generate output forces of up to 15N [80], [81]. In contrast, suturing requires a dexterous end-effector capable of holding tools and performing complex manipulations similar to the hands of a surgeon, with low interaction force and within a small workspace. Last but not least, the design of a surgical robot should consider the limited amount of space available in the operating room, and the possible interaction with other surrounding equipment, such as ultrasound probes and supporting structures used in procedures relying on intra-operative imaging.

These requirements motivated the design of a modular robotic platform based on a macro/micro unit architecture [82], [83] consisting of two decoupled robotic structures that can be controlled independently as well as in concert.

- A *macro unit* with 4 degrees of freedom (DOF) serves as a gross positioning unit, to position the micro-unit over the region of interest where the tool has to be placed. For this purpose, a 3-DOF linear delta robot [84] was selected, as this parallel kinematics offers a rigid platform capable of carrying the weight of the micro unit, while ensuring high stiffness and positioning accuracy. The three parallelogram arms of the delta structure are actuated by three linear spindle drives. An additional DOF on the moving platform controls the rotation of the micro-unit base to adjust its orientation during the surgical procedure. For sake of convenience and space constraints, the linear delta was attached to a custom-made 2-meter long table in the first prototype. However, the arms of the delta structure could easily be flipped upwards, allowing the linear drives to be fixed to the ceiling or a supporting structure for installation in an operating room.

- A dexterous *micro unit* capable of manipulating different surgical tools is mounted on the moving platform of the macro unit. The micro is based on hybrid kinematics and offers 4 DOF, mimicking the arm of the surgeon (shoulder flexion/extension, shoulder rotation, elbow flexion/extension and forearm pronation/supination). In the case of the puncturing procedure, the needle is mounted on the distal end of the robotic arm of the micro unit (Fig. 4.1). The first three DOF of the micro unit are used to orient the needle and are actuated remotely (from the moving platform of the macro-unit) while the fourth DOF allows rotation of the needle around its axis via a belt and pulley drive located behind the needle holder. The needle can be easily detached from its holder to allow the successive insertion of multiple needles. The needle holder incorporates a 6-axis force/torque sensor (ATI Nano 17, ATI Industrial Automation Inc., NC, USA) to measure interaction forces and torques during insertion. For more complex surgical tasks, such as suturing, the needle is replaced with a cable-actuated wrist module that provides 3 additional DOF (wrist flexion/extension, radial/ulnar deviation, and a gripper) and can be easily attached to the force/torque sensor. A second arm with the same DOF is further mounted on the moving base of the macro unit (Fig. 4.2 and Fig. 4.3), resulting in a versatile, bimanual robotic surgical platform with up to 18 DOF.

Position sensing is achieved through encoders located at the level of each actuator, and through potentiometers along the drives, providing a redundant position measure for safety purposes. The end-effector position is computed by solving the kinematics of both the macro and micro units independently, and then combining them.

The control architecture of the robotic surgical platform is organized in a hierarchical way. A low-level controller performs position and velocity control in a cascaded man-

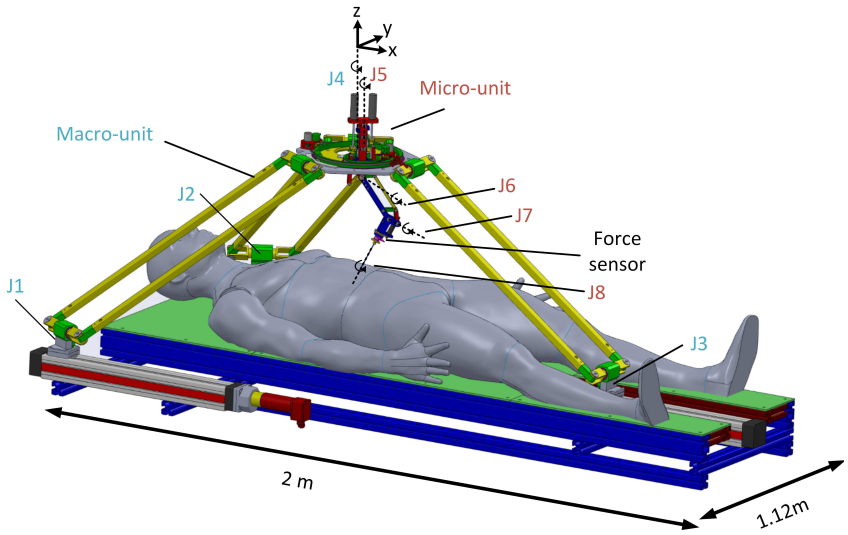


Figure 4.1: The I-SUR robotic platform for automatic needle insertion. The robot consists of a macro unit (linear delta with 4 DOF, J1-J4) for gross positioning, and a micro-unit (4 DOF, J5-J8) to hold and orient a cryoablation needle. The needle holder is attached via a 6-axis force/torque sensor.

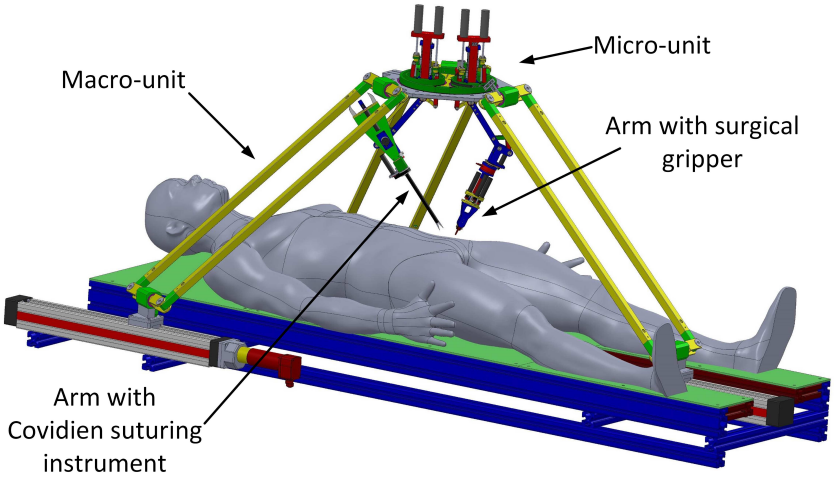
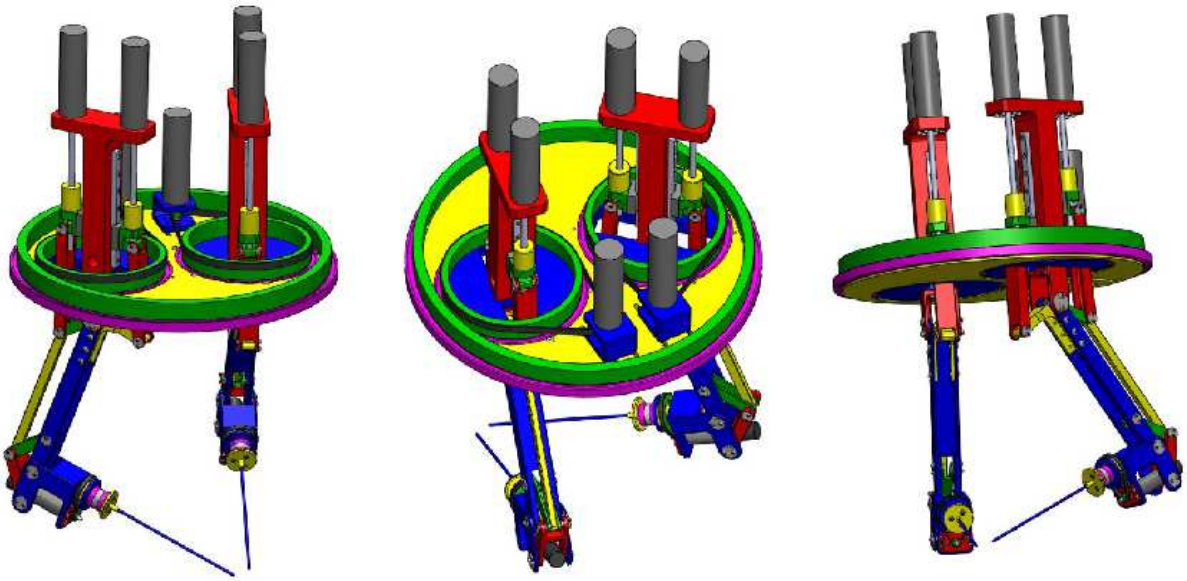
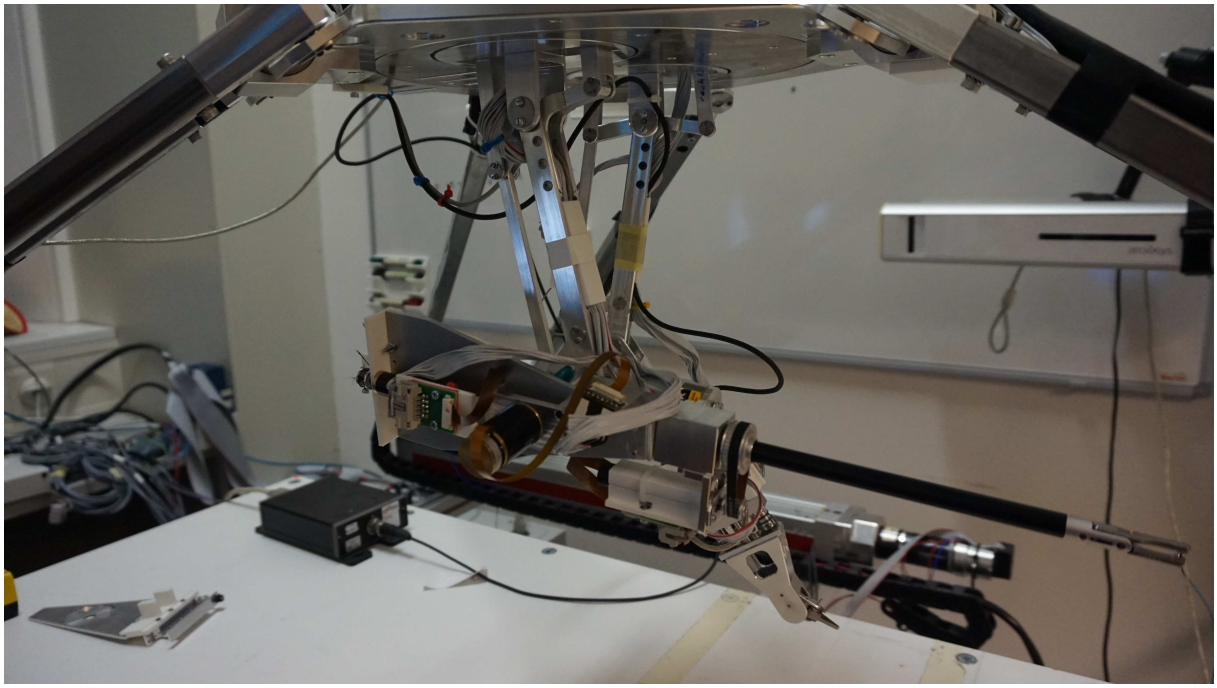


Figure 4.2: The I-SUR robotic platform for automatic suturing. The robot consists of a macro unit (linear delta with 4 DOF, J1-J4) for gross positioning, and two micro-units (4 DOF, J5-J8 and 6 DOF, J9-J14) to hold and orient a suturing tool (e.g. Covidien Endo Stitch) and a gripper.



(a) CAD model



(b) Real robot

Figure 4.3: Adaptation of the micro-unit for automatic suturing. Two arms are mounted on the moving base of the macro unit. Each arm can provide up to 7 DOF.

ner [85] at the joint-level, and is used for trajectory following. A high-level controller implemented on a separate PC generates the commands for the automatic execution of the surgical tasks, and includes the reasoning and cognitive processes required for such tasks as well as the interaction control schemes described in Chapter 2. The interaction control runs on top of the implemented joint position control, which allows the robot to dynamically interact with the environment in a safe and controlled manner.

The low-level control layer is implemented in real-time LabVIEW 2013 (National Instruments, USA), and runs on a PC with an 8-core Intel i-7 (3.4 GHz) processor. For optimal control of the multi-DOF robot, joint/velocity control runs at 10 kHz on an integrated field-programmable gate array board, while trajectory following is performed at 2 kHz on the PC.

4.3 Puncturing

Figure 4.4 shows the robotic platform for the needle insertion under US monitoring. The commercial robot UR5 holds the US probe: the sensing system detects on the US images the motion of the needle to guarantee the safety of the procedure. The I-SUR robot holds the needle and performs the puncturing according to the planned trajectory. During the needle insertion task, the US probe is first placed on the surface of the body, aligned with the expected needle tip trajectory, and then the needles are mounted on the robot end-effector and automatically inserted one by one.

For the puncturing tests, a kidney box phantom replicating the right side kidney of a human being and its surrounding structure has been used (Fig. 4.5). The phantom includes other artificial organs (liver, colon, ribs), produced using high-fidelity CAD models as described in [21], which are enclosed by a gelatin layer that replicates the features of human skin and fat. Two 20 mm tumors were added to the lower pole on the posterior face of the kidney. The surrounding space around the kidney, liver and colon was filled with water which allows the US-based inspection of inner structures of the phantom.

If an unexpected event occurs, the manual intervention of the surgeon is required. Thus, the setup includes a PHANTOM Omni haptic device that can be utilized as the master device when the system switches to teleoperation.

4.3.1 Planning of the surgical task

As introduced in Chapter 1, modern cryosurgery is frequently performed as a minimally-invasive procedure, with the application of hollow hypodermic needles (cryoprobes), strategically located in the area to be destroyed. The cryoprobes layout is a key factor for

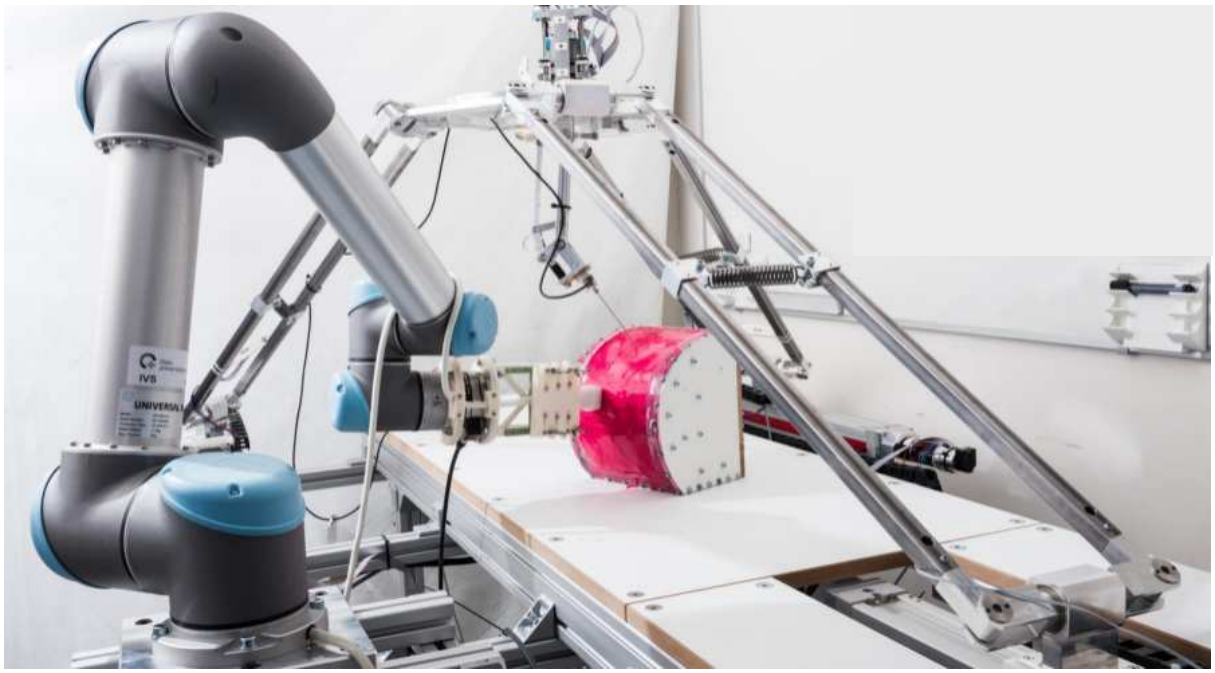


Figure 4.4: The overall robotic platform for needle insertion. On the left, the UR5 robot holding the US probe; on the right, the I-SUR robot holding the needle; on the operating table, the phantom of the human abdomen.

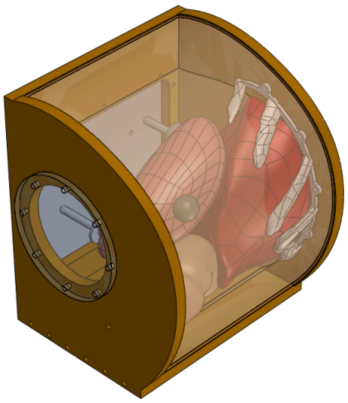


Figure 4.5: Phantom replicating the human abdomen. Kidney with two tumors in the middle of the phantom, part of the liver and the covering ribs can be seen on the right, part of the colon visible on the left.

achieving the highest overlapping between the frozen region and the target region, i.e. the tumor, and, therefore, for the success of the cryoablation procedure.

Each cryoprobe generates an iceball which is associated to a specific isotherm and the iceball size depends on the kind if probe being used (Fig. 4.6). Without loss of generality, in the following will be considered the IceRod type. However, the algorithm can be easily applied to other kind of cryoprobes.

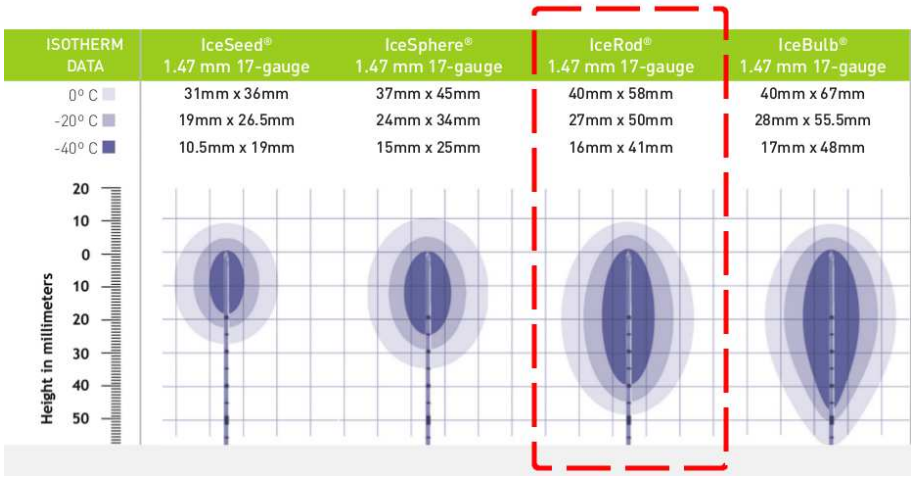


Figure 4.6: Iceball shape and size related to different cryoprobes types [86].

Since experimental models have shown tissue necrosis of cancer cells at temperature between -20 °C to -40 °C [16], in the following the iceball will be considered as an ellipsoid of size 27x50 mm, corresponding to the isotherm of -20 °C (Fig. 4.6) for IceRod needles. However, this choice is for demonstration purposes only; the planning algorithm presented in this section is independent of the value of the isotherm.

Building the model of the patient

Anonymized series of CT scans from a patient going through kidney tumor cryoablation were retrieved from the San Raffaele database to reconstruct part of the patient's abdomen for evaluating the iceball growth algorithm. The segmentation of the abdomen covered the right kidney and the surrounding structures and organs (i.e. part of liver, ribs, intestine and back-bone). Fat, muscles and skin were segmented as homogeneous layer. The models of the organs were segmented from the first series where one of the cryoablation needles was already inserted into the kidney tumor. This scan was chosen for good visibility of the tumor and for matching the CT scan series with the later phase CT scan series by the using the first needle position as a reference. The reconstruction of the organs was done by using open source software 3D Slicer [87].

The CT scan with the visible first needle was first cropped down (see Fig. 4.7) for faster segmentation process. The segmentation of the organs was done semi-automatically by using simple region growing segmentation algorithm and later repairing the results manually. The segmented layers were then assembled into 3D models by using Model Maker module in 3D Slicer.

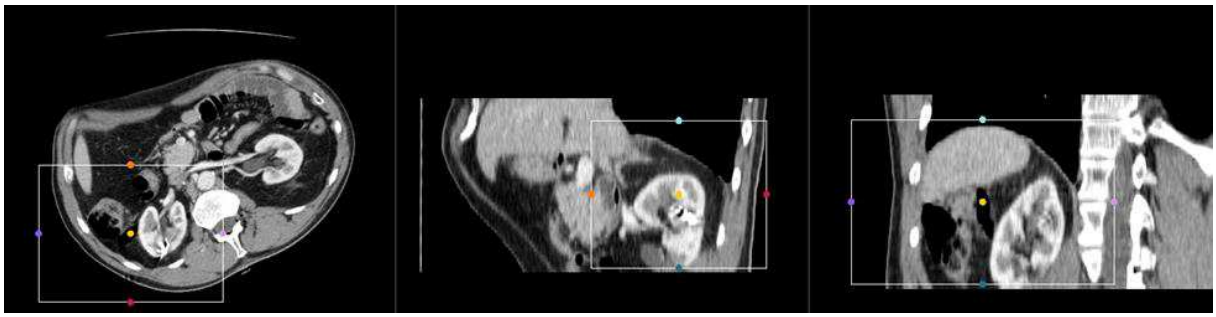


Figure 4.7: Cropped volume of the CT scan.

The segmentation of the iceball was done from a CT scan series with 4 visible needles with similar technique described before. Because of the movements of the patient throughout the procedure, the CT scans were misaligned. The scans were manually aligned (by changing the image origin of the 4-needle scan) according to the first inserted needle's tip and to the edges of the kidney. The model of the iceball aligned to other models can be seen (with blue color) in Fig. 4.8. Finally, two point coordinates of each needle were collected from the CT scan to reconstruct the same situation in the iceball growth simulation (see image above right lower corner for the visible needle points), as reported in the following.

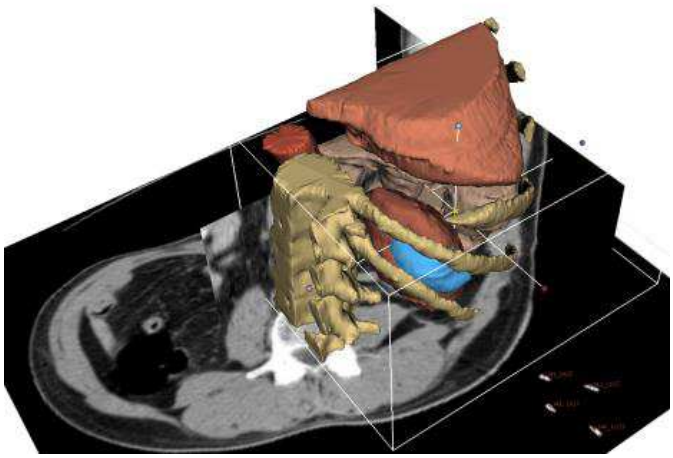


Figure 4.8: Models of liver, intestine, aorta, backbone and kidney with iceball from the CT data.

Computation of number and position of the needles: the *Cryo-planner*

Currently, the process of selecting the correct placement of the cryoprobes is an art held by the surgeon, based on his/her experience and rules of thumb. Cryoprobes are typically operated in a trial-and-error fashion, until the entire target volume is thought to be frozen.

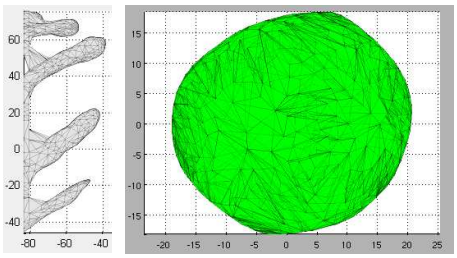
Suboptimal localization may

- leave regions in the target volume unfrozen
- lead to cryoinjury of healthy surrounding tissues
- require an unnecessarily large number of cryoprobes
- increase the duration of the surgical procedure
- increase the likelihood of post cryosurgery complications

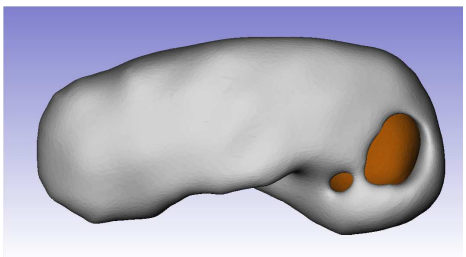
all of which affect the quality and cost of the medical treatment. Computerized planning tools would help to alleviate these difficulties. Few works have been done in this direction. In [88] Lung et al. developed an optimization technique, called the *force-field analogy*, in which heat transfer simulations are executed to move the cryoprobes into an optimum layout. Then, in [89] Tanaka et al. implemented the force-field analogy together with a technique called *bubble packing* [90]. These works do not take into account the obstacles that have to be avoided while inserting the needle, e.g. ribs or organs, or specific constraints on the insertion procedure. Furthermore, the number of iceball to be used is chosen a priori by the surgeon.

The essential task for a planning tool is to identify the number and the best locations for the cryoprobes to generate iceballs that completely freeze the target region, i.e. the tumor, while minimizing cryoinjury external to the target region. Furthermore, needle insertion is subjected to several constraints:

- *Forbidden regions.* Areas that the needle has to avoid (i.e. ribs and organs).
- *Tumor inserting area.* Surface of the tumor where needles can be inserted (healthy tissue of the kidney should not be damaged).
- *Maximum angle between needles.* Relative angle between needles should range from 0 to 20 degrees. However, the value for the maximum angle allowed can be chosen by the surgeon.
- *Insertion grid.* Needles can be inserted only through an adhesive grid applied on the skin.



(a) Points defining the ribs and the tumor surface



(b) CAD model of the phantom (kidney with tumor)

Figure 4.9: Input data for the planning algorithm

- *Needle collision.* Needles must not collide each other.

The algorithm proposed in the following requires as input the points defining the surface of the tumor and the eventual constraints. These points are extracted from the model of the patient body extracted following the procedure described above. In particular, the points are extracted from the MRI image of the kidney with the tumor and its CAD model (that can be easily obtained from the MRI data). Figure 4.9(a) shows the ribs and the tumor points extracted from the CAD model (Fig. 4.9(b)) of the model reconstructed. The system is completely configurable and every kind of obstacle can be taken into account, e.g. the Inferior Vena Cava or the suprarenal glands. Indeed, it is sufficient to provide to the algorithm the points defining the surface of the obstacle extracted from the MRI image, as for the kidney and the tumor.

The planning algorithm computes the number and the location of needles necessary to entirely cover the tumor, while minimizing the damage to the healthy tissue and satisfying all the possible constraints to needle insertion. The algorithm goes through the following steps (Fig. 4.10).

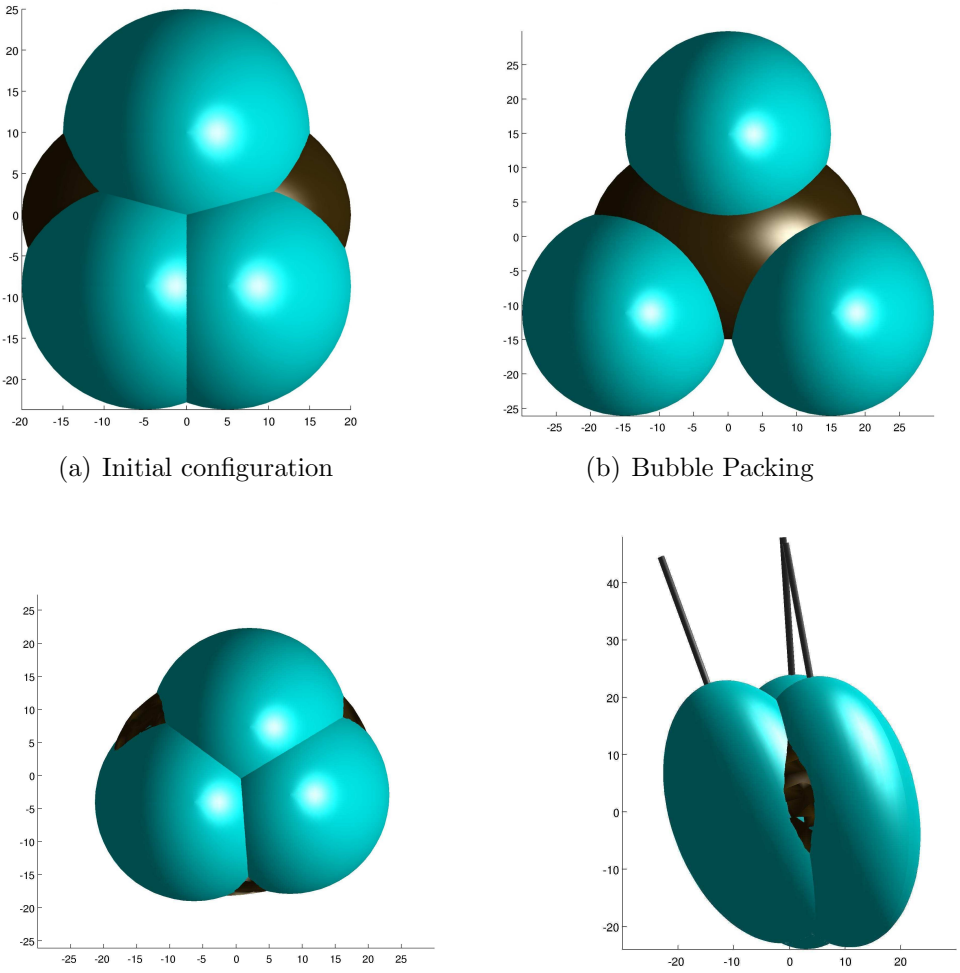
1. **Computation of the Initial Number of Iceballs**

   At the beginning, the algorithm uses an iterative procedure based on the tumor volume and dimension and its relation with the iceballs volume and obtains an approximation of the minimum number of iceballs necessary to cover the tumor.

   The initial positions of the centers of the iceballs are equally spaced along a circumference inscribed in the perimeter of the longitudinal section of the tumor.

2. **Bubble Packing**

   The initial configuration of the iceballs could present iceballs overlapping, as in Fig. 4.10(a), or an excessive distance among iceballs. In this situation the next steps of



(a) Initial configuration

(b) Bubble Packing

(c) Force-Field Analogy

(d) Orientation Optimization

Figure 4.10: Steps of the planning algorithm. Iceballs (blue) and tumor (brown).

the algorithm would be too time consuming.

To avoid this problem, Bubble Packing [91], [90] has been applied (Fig. 4.10(b)). It is a physically–based approach that efficiently finds an even distribution of an arbitrary number of points inside a given geometric domain. The method first generates spherical elements, called *bubbles*, inside the domain, then defines van der Waals–like forces between bubbles. With this proximity-based force, two adjacent bubbles attract each other when they are too far apart, and repel each other when too close. The objective of the Bubble Packing phase is to evenly distribute the cryoprobes, in order to shorten the optimization process in the following phases of planning.

Since in this phase only bubbles translation is performed, Bubble Packing can be extended to ellipsoidal bubbles, i.e. the iceballs generated by the cryoprobes, instead of spherical bubbles. Further details on Bubble Packing can be found in [91], [90].

3. **Force–Field Analogy**

   The third phase of the planning algorithm consists of a modified version of Force–Field Analogy suggested in [88]. This phase aims at finding the optimal position of the iceballs in order to completely cover the tumor, considering only iceballs translation. It is required not to damage healthy tissues over 10 mm out of the tumor contour.

   At the beginning, tumor, iceballs and the surrounding region are discretized into a set of points, named *defective points*. Each point applies an attractive or repulsive force on the iceballs to take them into an optimal configuration.

   The points that apply a **repulsive force** are classified into the following types.

   - *External defects* (T_E). Points located outside the tumor but inside the iceballs representing surrounding healthy tissues that would be wrongly cryoinjured.
   - *Contour defects* (T_C). Points located into a shell 10 millimetres thick from the tumor contour and inside the iceballs representing the limit for healthy tissue damage.

   Both the external and the contour defects apply a repulsive force on the center of the iceball in order to move the cryoprobes from the external region towards the center of the tumor.

   - *Superposition defects* (I_I). Points located inside the intersection area between bubbles representing iceballs superposition. These defects apply a repulsive

force in order to prevent gathering too many cryoprobes at the same locations.

The points that apply an **attractive force** are clustered into the following categories.

- *Internal defects* (T_I). Points located inside the tumor but outside the iceballs representing regions in the tumor that would not be cryoinjured.
- *Surface defects* (T_S). Points located on the surface of the tumor that are not covered by the iceballs.

Both the internal and the surface defects apply an attractive force on the iceball in order to attract the cryoprobes towards the areas that are not covered.

The defective points are used to directly drive the cryoprobes location. In fact, the iceball translation is performed proportionally to the total force applied to a cryoprobe, computed as

$$F_{TOT} = \sum_{i \in \mathcal{P}} W_i d_i^2 \tag{4.1}$$

where $\mathcal{P} = \{T_E, T_C, I_I, T_I, T_S\}$ is the set of defective points types, d_i is the distance between the center of mass of the defective points of type i related to a given iceball and the center of the iceball and W_i is an experimentally determined weight for defects of type i. For defects that apply an attractive force, the weight is positive, while for defects applying repulsive forces it is negative.

Since the total force is proportional to the distance d_i, the displacement of an iceball far away from the tumor is greater than the displacement of an iceball closer to the tumor. The iceball translation is performed only if the movement allows to reduce the value of the following objective function.

$$f = \left| \sum_{i \in \mathcal{P}} N_i W_i \right| \tag{4.2}$$

where N_i is the number of defects of type i.

The Force-Field phase stops when further iceballs movements would increase the value of the objective function (4.2). Because of the simple way the initial number of iceballs is computed, it may happen that at the end of this phase the tumor is not completely covered (i.e. $N_{T_I} > 0$ or $N_{T_S} > 0$). The resulting iceball configuration (Fig. 4.10(c)) represents the best solution considering only iceball translation.

At this point, the algorithm determines if it would be worth pursuing the Orientation Optimization by analysing the amount of uncovered surface. If not, the algorithm adds a further bubble and goes back to Phase 1.

If the tumor is completely covered, the algorithm goes through the initial part of the Orientation Optimization, where the constraints satisfaction is verified. If all the constraints are satisfied, then the algorithm ends. Otherwise, the algorithm continues by changing the orientation of the iceballs for satisfying the constraints.

4. **Orientation Optimization**

   The goal of this phase is to complete the tumor coverage, while satisfying the constraints to needle insertion previously introduced. The needle is computed as the straight line passing through the center of the iceball and the tip. The length and the diameter of the needle are given by the specifics of the cryoprobe.

   The initial collisions between needles or between a needle and an obstacle are solved rotating the needle(s). Each step executed to solve a constraint is performed to reach the configuration with minimal differences compared with the one previously obtained. To this aim, the involved needle is rotated step-by-step towards sequence of points generated on the minimal distance line from the intersection point.

   If the output of the Force-Field Analogy is to puncture a forbidden area on the kidney or on the tumor, the needle is rotated until a non-forbidden region is reached. In this phase, the defects are no longer considered as single points but as even regions.

   If the iceballs configuration satisfies the constraints, the algorithm searches the defective regions which have not been covered in the previous phase, as shown in Fig. 4.11. Then, the algorithm computes the center of mass for each defective

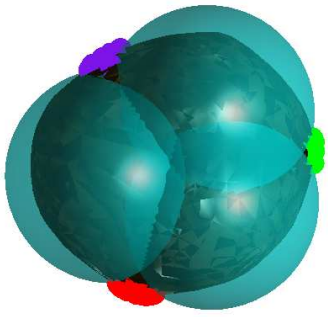


Figure 4.11: Detection of the defective regions

region and its distance from each iceball's center. This distance is used to establish which iceball has to be rotated and the rotation direction. The rotation is performed towards the center of mass just computed, on the straight line connecting the iceball

top and the center of mass. While the iceball is rotating, the algorithm constantly checks the constraints and introduces alternative solutions if one of them is wrong. Once the constraints have been satisfied, the tumor coverage is verified: if the covered part of the tumor is smaller than in the previous iteration, the last rotation is removed.

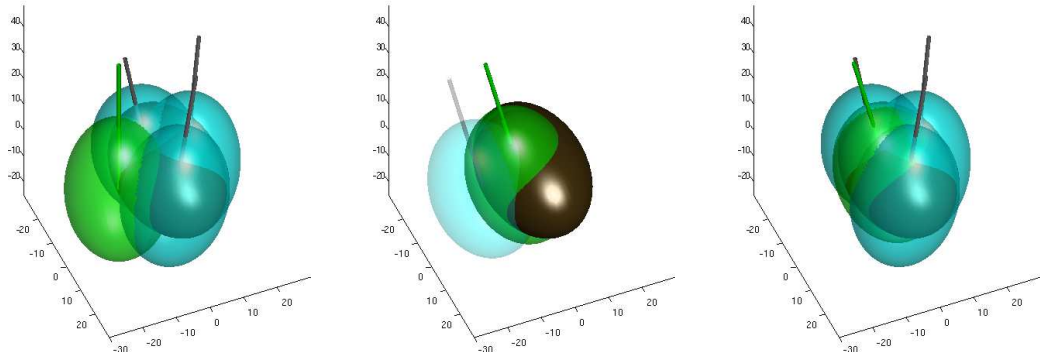
The Orientation Optimization stops when the tumor is completely covered or when there are no more convenient solutions (Fig. 4.10(d)).

5. **Adding Iceball**

The Orientation Optimization phase provides a possible solution considering the given iceball number and the imposed constraints. If the constraints are too restrictive or the tumor has a particular conformation hard to deal with, it is possible that at the end of the previous phase some parts of the tumor are not covered, as in Fig. 4.10(d).

Since the total tumor coverage is mandatory, an additional iceball is required. The location of the further iceball depends on the number and position of the uncovered regions.

If there is only one uncovered tumor region, the new iceball is inserted with the center coincident with the center of mass of the defective area. The iceball is then rotated and translated towards the center of the tumor, in order to minimize the damage to the outer healthy tissues (Fig. 4.12). The translation continues until a part of tumor becomes uncovered again or any constraint is broken.



(a) The iceball is inserted into the center of mass

(b) The iceball is rotated and translated

(c) Iceball final position

Figure 4.12: Optimization of the position of the new iceball

If the defective regions at the end of Orientation Optimization are more than one, it

is hard to find a position for the added iceball which would not violate the imposed constraints. However, the algorithm tries to find a solution to achieve the final goal. The new iceball is inserted in correspondence of the center of mass of all the defective regions and rotated towards the nearest defective area, always checking the constraints satisfaction.

If the additional iceball cannot complete the tumor coverage with the technique here reported, or if some constraint is not satisfied, the algorithm goes back to Phase I, and run again including the further iceball.

The final result of the planning algorithm is an output file containing, for each needle, the inserting pose (position and orientation) on the skin and the target pose on the tumor.

Simulation of the iceball growth

A heat transfer model of the iceball growth has been computed in order to simulate and verify the effective coverage of the iceballs coming from the planning algorithm. The model has been made starting from the thermal energy balance for perfused tissue and from Pennes' bioheat equation [92].

In order to easily solve the heat transfer PDE, a finite-difference method has been chosen for approximating the spatial derivatives, i.e. a first order forward difference for the first term and a second order central difference for the second one. The continuous function then turned into discrete within a 3D grid with a sub millimeter resolution. Body temperature has been chosen as initial condition, while the temperature profile of the external surface of each probe has been set as boundary condition. To validate this approach, it has been chosen to match the 3D model reconstructed from intra-operative imaging described above with the simulation of iceball growth taking as initial positions the cryoprobes displacement obtained from the same CT dataset, discretized and adapted in the computation grid. Even though the -40 °C isotherm is the one interesting for planning purposes, the validation protocol has taken into account just the 0 °C line because its compatibility with CT data. The models are comparable in terms of dimension and shape, as can be seen in Fig. 4.13.

4.3.2 Software architecture

Developing and experimenting a full case study for a complex engineering task, such as the one addressed above, require efforts from different research teams and, therefore, the integration of possibly different software technologies. In fact, reuse of previously

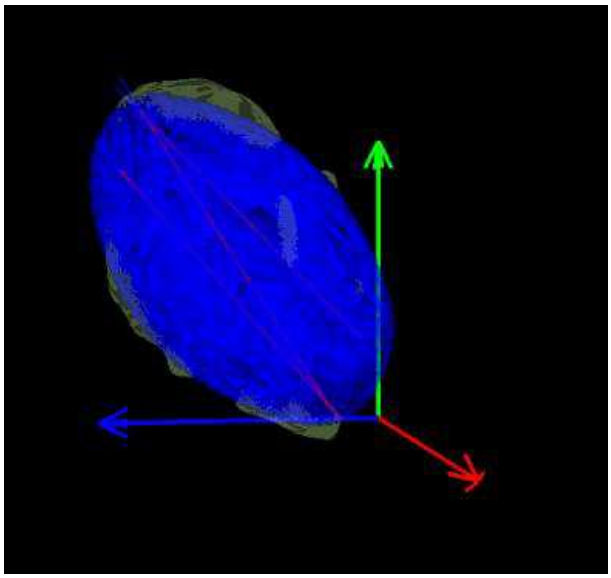


Figure 4.13: Superimposition of simulated iceball (blue) on real CT data (green). Cryoprobes are visible inside (red).

developed software is exploited first, in order to speed up the setup of planned experiments, even though the tools, libraries and platforms used to develop such software are not homogeneous. Moreover, some hardware devices (e.g. the PHANTOM Omni included in the experimental setup for teleoperation) can be more easily integrated using specific operating systems (e.g. MS Windows), because of the stability of the drivers for that platform. Therefore, the core part of the robotic control system is implemented using the Orocos component-based approach described in Chapter 3 and runs on PCs with Ubuntu 12.04 Linux operating system and Xenomai hard-real time extension, but the complete software setup adopted for experiments integrates some parts not Orocos-based. This aspect further highlights the flexibility of the proposed software architecture, whose full deployment for the considered case study is shown in Fig. 4.14.

The software running on the PCs with real-time Linux is entirely made of Orocos components. Even though the focus is on the reconfigurability and extensibility of the overall architecture, rather than on the performance of its implementation, the Orocos deployment on Xenomai allowed to safely execute the high-level control loop depicted in Fig. 3.7 and Fig. 3.8 at a sampling frequency of 1 kHz. The real-time control components communicate at intra-host level with the lock-free, thread-safe mechanisms embedded in Orocos RTT, while inter-host communication is allowed by Orocos native support for CORBA. An interesting feature of the Orocos CORBA extension is the possibility to define custom *typekits* allowing the exchange of non-standard data types on a CORBA interface, which was exploited to connect Data Ports with KDL types representing robotic and geometric

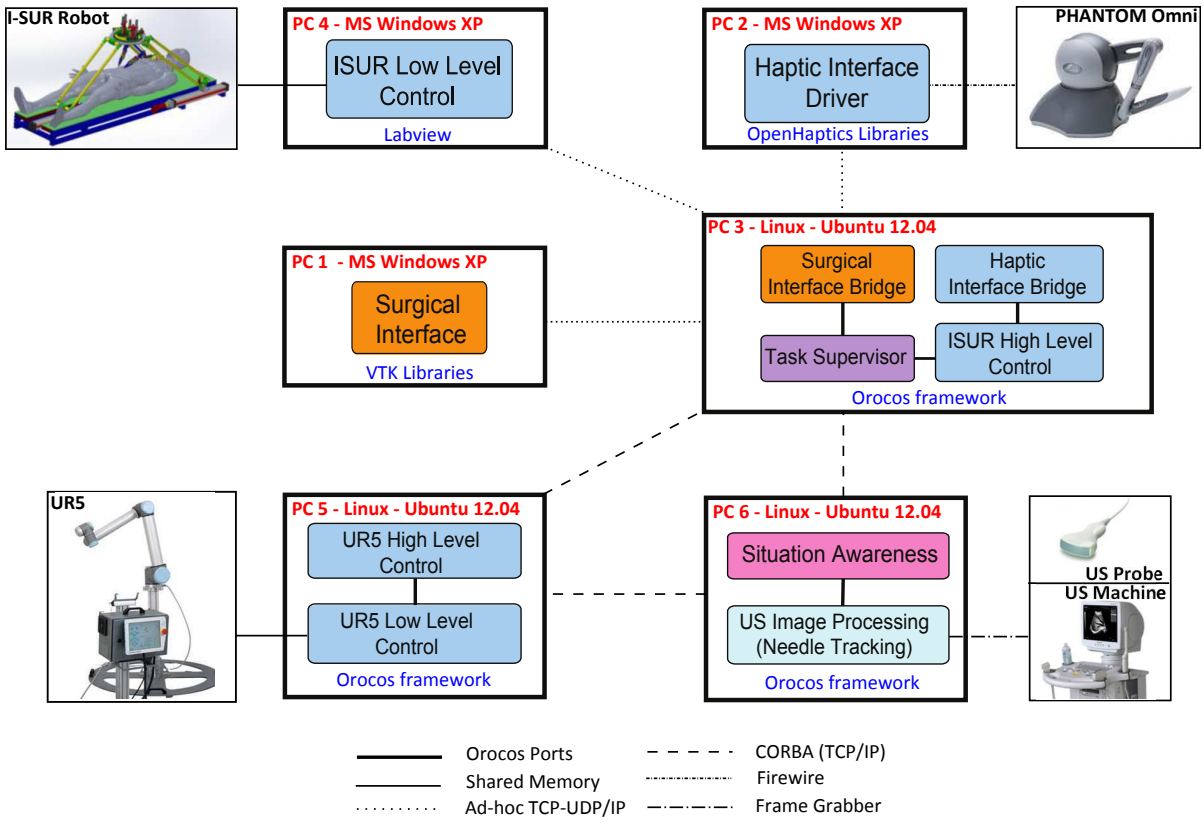


Figure 4.14: Distributed deployment of the component-based software architecture for autonomous surgical robots performing the puncturing task.

primitives. The other PCs adopt MS Windows XP as operating system, because of the mentioned limitation of PHANTOM Omni device drivers, the fact that the prototype of the Surgical Interface has been developed using MS Visual C# language and the fact that the low-level control of the I-SUR robot, receiving Cartesian pose commands from the Variable Admittance Controller, is implemented in National Instruments LabVIEW 2013. The latter is also interacting with an integrated FPGA (*Field-Programmable Gate Array*) board, executing joints position/velocity control at 10 kHz sample rate.

The Task Supervisor can coordinate the full system, since CORBA interoperability can be used to exchange events and execute deployment and reconfiguration of remote components (e.g. set properties, request operations, start or stop components activity). The interaction with non-Orocos parts is allowed by the development of specific *bridge* components, embedding socket-based exchange of TCP or UDP packets on a standard Ethernet connection. Even though CORBA interfaces could be (and will be in future works) developed also within MS Visual C# or LabVIEW environments, plain TCP/IP protocol was preferred to speed up integration and reduce overheads.

Remark 5. *Neither CORBA (on a TCP/IP transport layer) nor plain TCP/IP theoretically guarantees deterministic real-time behavior, especially if MS Windows nodes are part of the network. However, a careful choice of network topology and the use of high-speed switches allow to obtain satisfying communication latencies, as described in [93] and [94].*

The first experiment performed on the proposed robotics setup is described by the UML Sequence Diagram of Fig. 4.15 and aims at showing the interesting features of the software architecture. The results about the control part and the switch to teleoperated mode will be treated in Section 4.3.3.

The robot autonomously performs the insertion of the needle into a phantom of the human abdomen. As described above, the phantom includes artificial organs, produced using high-fidelity CAD models, which are enclosed by a gelatin layer that replicates the features of human skin.

As shown in Section 3.5, the control software was developed using the Orocos component-based framework and includes, in addition to the computational components implementing the ***control strategies*** described in Chapter 2, a ***Task Supervisor***, implemented as an rFSM (*restricted Finite State Machine*) module [57], a ***motion planner***, which generates online collision-free paths in robot workspace coordinates using OMPL [58], ***communication bridges*** towards the low-level robot controller and a ***user interface*** interacting with the surgeon. The surgeon is in fact the primary driver of the surgical procedure, providing commands to proceed or interrupt the task execution sequence.

The diagram in Fig. 4.15 shows the system components, their timeline from top to bottom and exchanged commands, events or operation requests The timeline of the Task Supervisor also shows the states of the corresponding state machine, whose transitions are triggered by events denoted with the `e_` prefix. The surgical procedure is started when the I-SUR robot is in a *home* position. Then, the robot is moved along a collision-free trajectory, generated by the Trajectory Generator according to the path received by the Motion Planner, to a position allowing the surgeon to mount the needle onto the end-effector. The system is then required to wait an acknowledgment from the surgeon, verifying the correct installation of the needle. Once the needle is mounted, the robot moves towards the final target in three steps: first a collision-free trajectory to an approach position is generated and executed, then a sequence of two properly aligned linear trajectories of specified lengths are executed, without any collision-checking (i.e. no path request to the Motion Planner), to get in contact with the skin and then penetrate it. The robot is finally stopped and waits for the removal of the needle from its end-effector while the needle is still inserted, in order to simulate the cryoablation procedure. However, the experiment ends with the simulation of an undesired needle bending, detected by the US image processing of the Sensing/Reasoning module, that causes the surgeon decision to switch into teleoperated mode. The features of the switch into teleoperation will be described Section 4.3.3.

In the considered scenario, three interesting situations, in which the reconfigurability feature of the proposed Orocos-based architecture is exploited, can be highlighted:

1. **Scheduling reconfiguration:** for example, the behavior of the Trajectory Generator component changes from aperiodic to periodic (with a configurable sample time), whenever the `startMove` Operation is called to trigger the initialization of actual real-time trajectory generation (steps 6, 14, 18 and 21 in Figure 4.15). Another example of scheduling reconfiguration is given by the behavior of the bridge components (e.g. Surgical Interface Bridge), which changes from a low frequency periodic behavior, if no client is connected to the managed TCP-UDP/IP socket, to a high frequency periodic behavior, when the socket is properly connected.

2. **Properties reconfiguration**: as an example, the Variable Admittance Control Orocos component has a `stiffness` Property that is modified to adapt the robot interactive behavior, making it stiffer, just before the actual insertion of needle within the body/phantom starts (i.e. step 15 in Figure 4.15). Another component whose properties are reconfigured according to the task state is the Motion Planner. In particular, the set of obstacles to be avoided by the generated path can be changed at any time, so that interaction and contact with a particular object is

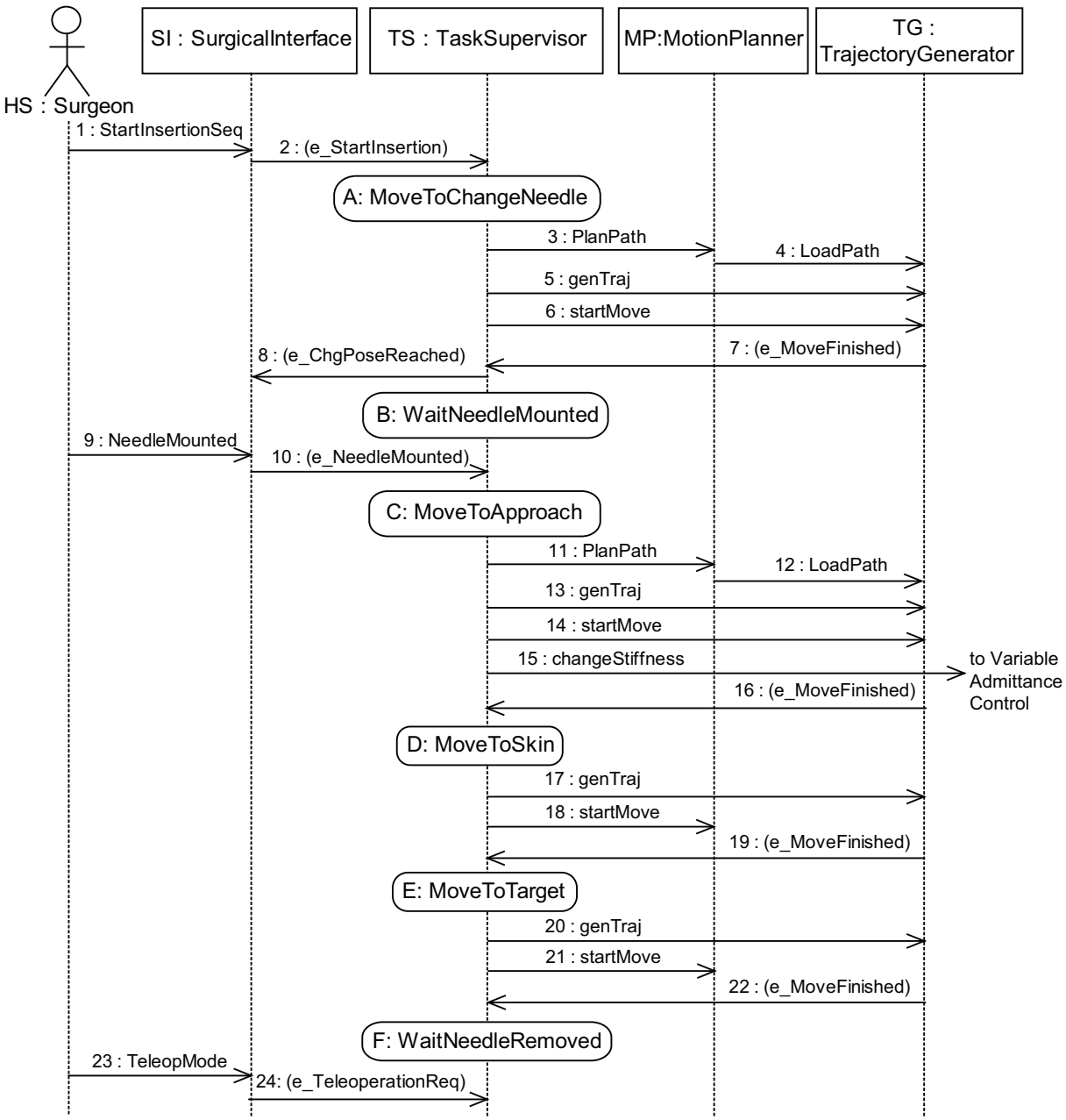


Figure 4.15: UML Sequence Diagram describing the first experimental scenario.

allowed during a given operating phase.

3. **Components interconnection reconfiguration:** during the autonomous/teleoperated mode switching triggered by the last step in Figure 4.15 the Motion Planner and Trajectory Generator components are stopped, while the Transparency and Passivity Layer components are started by the Task Supervisor. All the connections of Data Ports for the involved components are updated accordingly, as described by Figure 3.7 and Figure 3.8.

The features of the proposed control architecture are particular interesting for the surgical context addressed by the experiments, but are also applicable to other generic robotics and control applications. More precisely:

- The design process, strongly driven by requirements and oriented towards validation, is adequate for any control application related to a safety critical system.
- Networked control systems are more and more used in both industrial and research domains. The proposed component-based architecture handles the issues related to the use of multiple platforms and protocols thanks to reconfigurable bridge components, that could be extended to support the interoperability mechanisms used by industrial controllers.
- Different operating modes (e.g. *manual/automatic*, *production/service*, etc.) with different control configurations, as managed by the proposed architecture, are commonly required also by the automated systems of the manufacturing domain.
- Industrial robots executing grinding, polishing or similar operations (i.e. requiring contact force regulation) could benefit from the implementation of interaction control algorithms with variable parameters or collision-avoiding motion planners with reconfigurable sets of obstacles.

4.3.3 Experimental validation of the control strategies

A second scenario, similar to the previous, is prepared to validate the theoretical findings presented in Chapter 2 and highlight its features. The robot autonomously performs the insertion of the needle into the phantom. While the needle penetrates the phantom, the occurrence of an undesired event is simulated: the needle is assumed to miss the tumor and then has to be extracted and reinserted under manual guidance (teleoperation). To this aim, when the occurrence of the event is detected, the system switches to the teleoperated mode and the user can manually extract the needle from the skin and correctly reinsert it.

A PHANTOM Omni haptic device was utilized as the master device of the teleoperation system. Since the orientation DOFs of the PHANTOM Omni device are not actuated and, therefore, the reaction moments cannot be reflected to the user, the orientation is kept fixed when the system switches to the teleoperated mode. However, this practical limitation does not affect the generality of the approach described in Section 2.4.

The control software is the same described in Section 4.3.2 and the UML Sequence Diagram is shown in Fig. 4.16 where different features are highlighted respect to the one in Fig. 4.15.

Autonomous Needle Insertion

In this paragraph the focus will be on the first part of the experiment, where the needle is autonomously inserted by the robot following the desired trajectory to hit the tumor inside the phantom. Thus, the results of the implementation of the variable admittance control will be shown.

After many experimental evaluations, the inertial and damping parameters of the desired interaction model (2.19) were empirically chosen as the following constant diagonal matrices:

$$\begin{aligned} \Lambda_d &= diag\{0.5, 0.5, 0.5, 0.01, 0.01, 0.01\} \quad [Kg] \\ D_d &= diag\{D_{d1}, D_{d2}\} \end{aligned}$$

where

$$\begin{aligned} D_{d1} &= diag\{50, 50, 50\} \quad \left[\frac{Ns}{m}\right] \\ D_{d2} &= diag\{10, 10, 10\} \quad \left[\frac{Nms}{rad}\right] \end{aligned}$$

Since during the puncturing task the robot has to behave in different ways depending on the environment it has to interact with, the entries for the stiffness matrix change during the operation. Indeed, for example, the robot can be compliant while it is in free motion, while it has to be stiff for penetrating the skin.

To preserve the clarity of the presentation, the following plots will show only the results regarding the translational coordinates x, y and z. Similar results have been obtained for the rotational coordinates.

The evolution over time of the variable part $K_v(t)$ of the stiffness matrix is shown in Fig. 4.26, while the constant part is chosen as

$$K_c = diag\{K_{c1}, K_{c2}\}$$

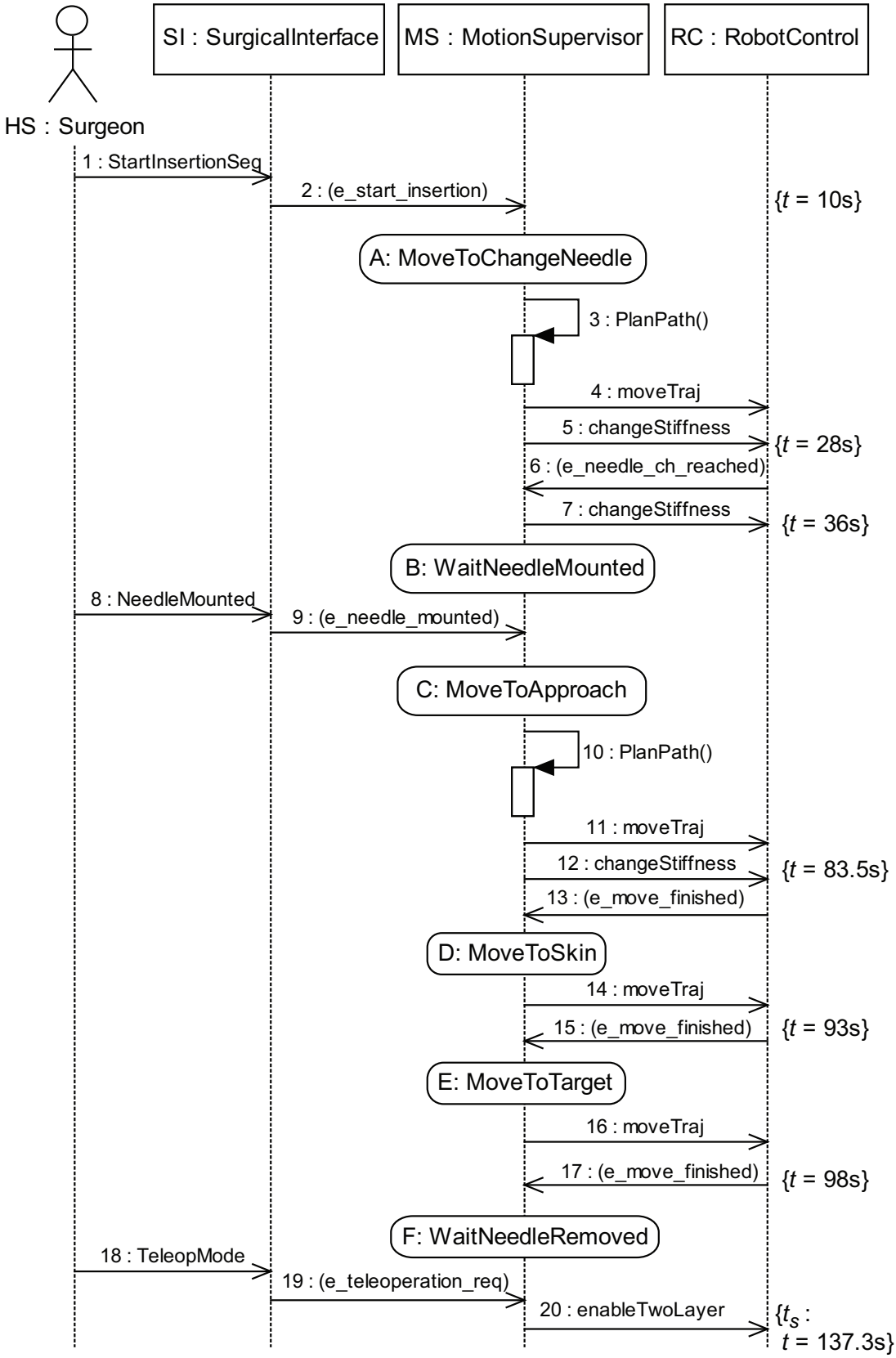


Figure 4.16: UML Sequence Diagram describing the second experimental scenario.

where

$$K_{c1} = diag\{10, 10, 10\} \quad \left[\frac{N}{m}\right]$$
$$K_{c2} = diag\{10, 10, 10\} \quad \left[\frac{Nm}{rad}\right]$$

To demonstrate that the system remains stable despite the stiffness changes, we even consider different ways of variating the stiffness profile. For example, during the movement of the robot to the position of needle change, the stiffness is augmented gradually, whereas when the robot is waiting for the needle to be mounted, the stiffness is changed instantly.

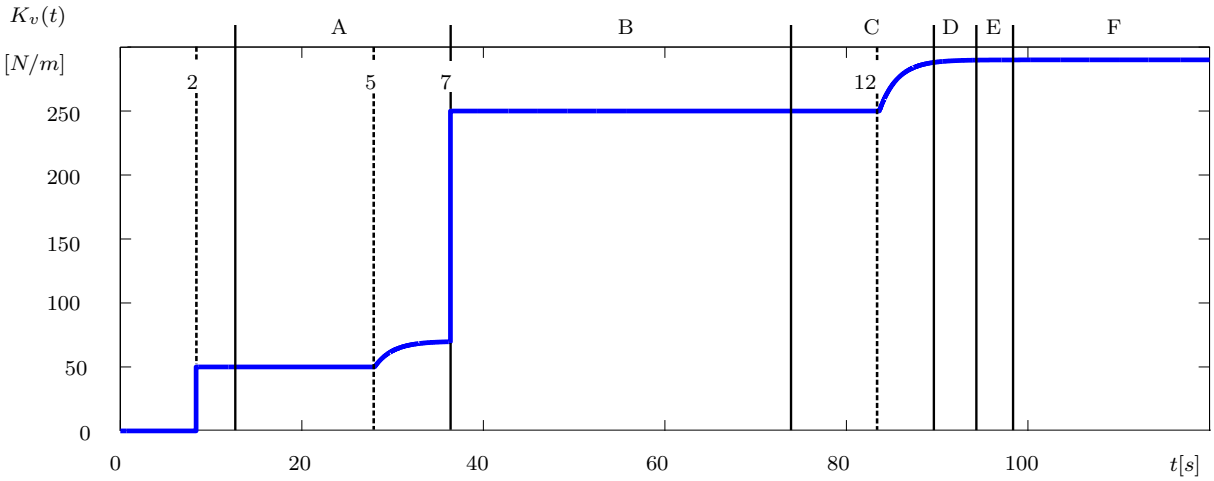


Figure 4.17: Evolution over time of the values chosen as diagonal elements for the variable part of the stiffness matrix during the autonomous needle insertion. (The numbers and the letters refer to the states and transitions in Fig. 4.15).

The desired Cartesian translational positions computed by the admittance controller are reported in Fig. 4.18. As expected, the commanded motion does not diverge over time and the system remains stable despite the many changes of stiffness. Figure 4.19 shows that the tracking error during the insertion of the needle (phase E) is below the acceptable value of 0.0011 m, thanks to the high values of the stiffness in this phase.

Figure 4.20 shows the behavior of the tank energy T. The energy thresholds are chosen as $\bar{T} = 10\ J$ and $\varepsilon = 0.1\ J$. When the stiffness changes instantly, at $t = 10\ s$ and $t = 36\ s$ (yellow regions in Fig. 4.20), the required energy to implement this behavior is extracted from the tank. On the other hand, the gradual changes of stiffness, red areas at $t = 28\ s$ and $t = 83.5\ s$, are dissipative actions, and thus the tank energy rises. The energy level has a big increment when the needle penetrates the skin ($t = 93\ s$) since this action is dissipative, but then, during the motion of the needle inside the phantom, the tank is

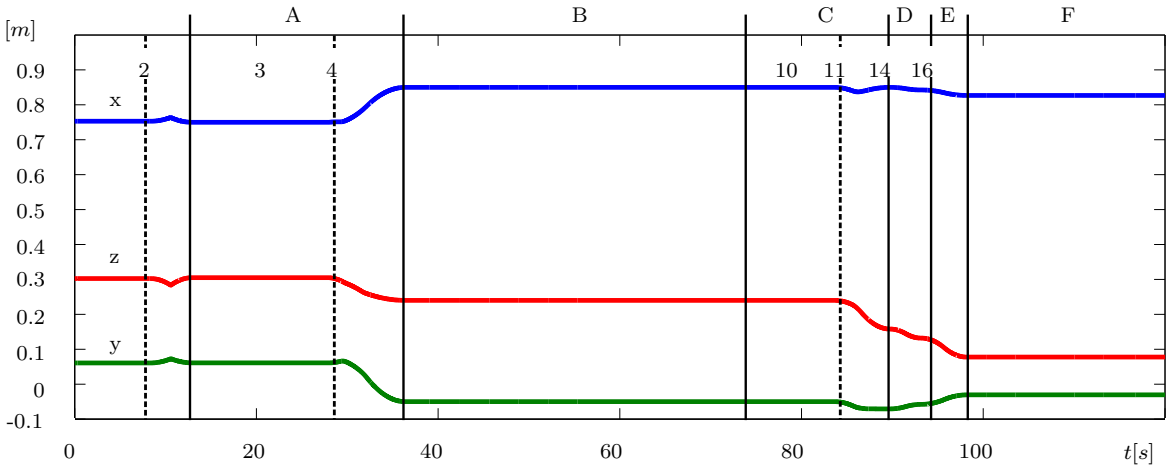


Figure 4.18: Desired cartesian positions computed by the admittance controller. (The numbers and the letters refer to the states and transitions in Fig. 4.15).

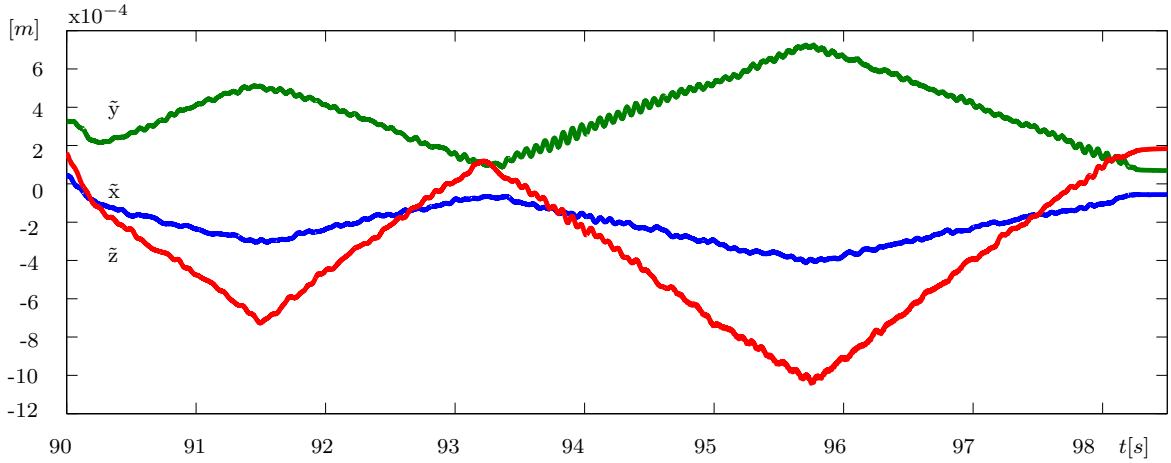


Figure 4.19: Tracking error (i.e. difference between the position of the robot end-effector and the desired trajectory) during the approach and insertion phases.

emptied again to perform the movement and the tank energy decreases accordingly (green region).

Little energy is extracted or inserted into the tank because, as shown for example in Fig. 4.19, the trajectory error $\tilde{x}(t)$ is small and then, from (2.27), the values of $w(t)$ used in (2.26) are small too.

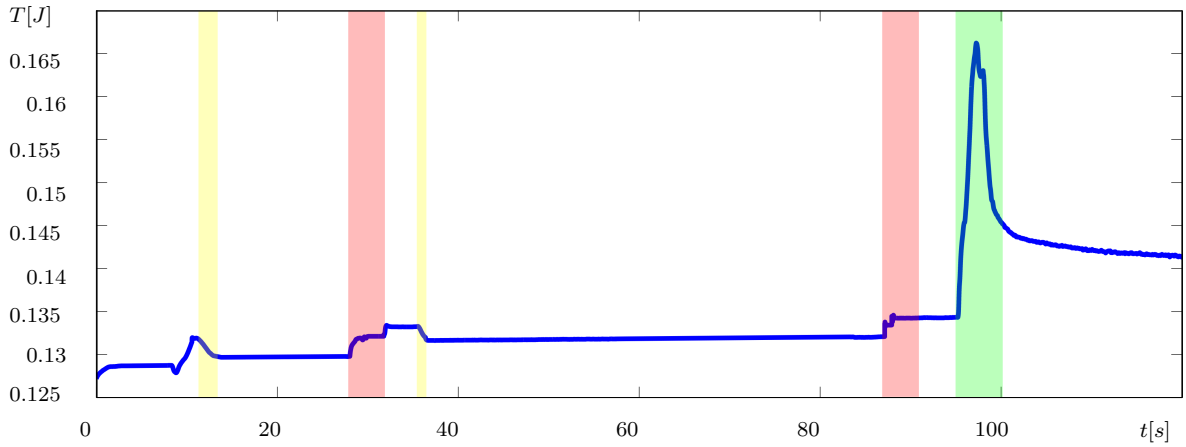


Figure 4.20: Evolution over time of the energy level of the tank during the autonomous insertion. Yellow regions correspond to instantaneous variations of stiffness, while red areas correspond to gradual changes of stiffness. The green region highlights the evolution of the tank energy during the motion of the needle inside the phantom.

Switching to Teleoperation for Manual Needle Extraction and Reinsertion

In this paragraph the second part of the experiment will be considered: the needle has been autonomously inserted, but for some unexpected event the tumor is missed. The system recognizes this event and commands to switch to teleoperation. The switching instant is $t_s = 137.3$ *sec*.

According to the theory developed in Chapter 2, the following design choices were made:

- the control parameters in Fig. 2.3 are:

$$K = 50 \ N/m$$

$$B = 1.1 \ Ns/m$$

$$R_m = R_s = 5 \ Ns/m$$

- the following values for the energy thresholds have been selected:

$$\bar{T}_i = 20 \ J$$

$$^{i}T_{ava} = 10\ J$$

$$^{i}T_{req} = 5\ J$$

$$\varepsilon_i = 1\ J$$

where $i = m, s$

- the functions $f_1(e(t))$ and $f_2(\lambda(t))$ in (2.51) and (2.52) are defined as

$$f_1(e(t)) = \frac{1}{2}\left[\cos\left(\pi\frac{e(t) - \bar{e}_1}{\bar{e}_2}\right) + 1\right] \tag{4.3}$$

$$f_2(\lambda(t)) = \frac{1}{2}\left[\cos\left(\pi\frac{\lambda(t) - \bar{\lambda}_1}{\bar{\lambda}_2}\right) + 1\right] \tag{4.4}$$

- the thresholds related to the functions $\alpha_1(e(t))$ and $\alpha_2(\lambda(t))$ are chosen as:

$$\bar{e}_1 = 0.005\ m$$

$$\bar{e}_2 = 0.01\ m$$

$$\bar{\lambda}_1 = 0.005\ m$$

$$\bar{\lambda}_2 = 0.01\ m$$

In Fig. 4.21 the Cartesian positions of master and slave are shown, while Fig. 4.22 shows the detail of the z coordinate of master and slave positions during the alignment transient. As shown in Fig. 4.21, at the switching instant master and slave are not aligned and the following pose offsets arise when teleoperation is switched on:

$$\begin{aligned} L_x(t_s) &= \mathrm{x_m}(t_s) - \mathrm{x_s}(t_s) = -0.006\ m \\ L_y(t_s) &= \mathrm{y_m}(t_s) - \mathrm{y_s}(t_s) = -0.01866\ m \\ L_z(t_s) &= \mathrm{z_m}(t_s) - \mathrm{z_s}(t_s) = -0.07735\ m \end{aligned}$$

According to the method described in Section 2.4, the rest length L is replaced by the continuous time function $l(t)$ whose initial value is given by L but at the end of the compensation its result is equal to 0. At the switching instant, the operator feels a force guiding him/her to align the master position with the slave position. At the same time, the functions $\alpha_1(e(t))$ and $\alpha_2(\lambda(t))$ defined in (2.51), (2.52) increase, because $l(t)$ and the misalignment decrease, respectively. The torque due to the bilateral coupling is applied to the slave device weighted by $\alpha_1\alpha_2$. This means that the slave would not move until α_1 and α_2 will be close to one.

At the end of the alignment phase, $\alpha_1 = \alpha_2 = 1$. This means that the teleoperation is completely operative and master and slave are fully coupled. Indeed, as shown in Fig. 4.22, the slave starts moving only when the master position is very close to the slave position, i.e. the distance between master and slave is below the threshold of 0.01 m, and the coupling between the two motions becomes effective.

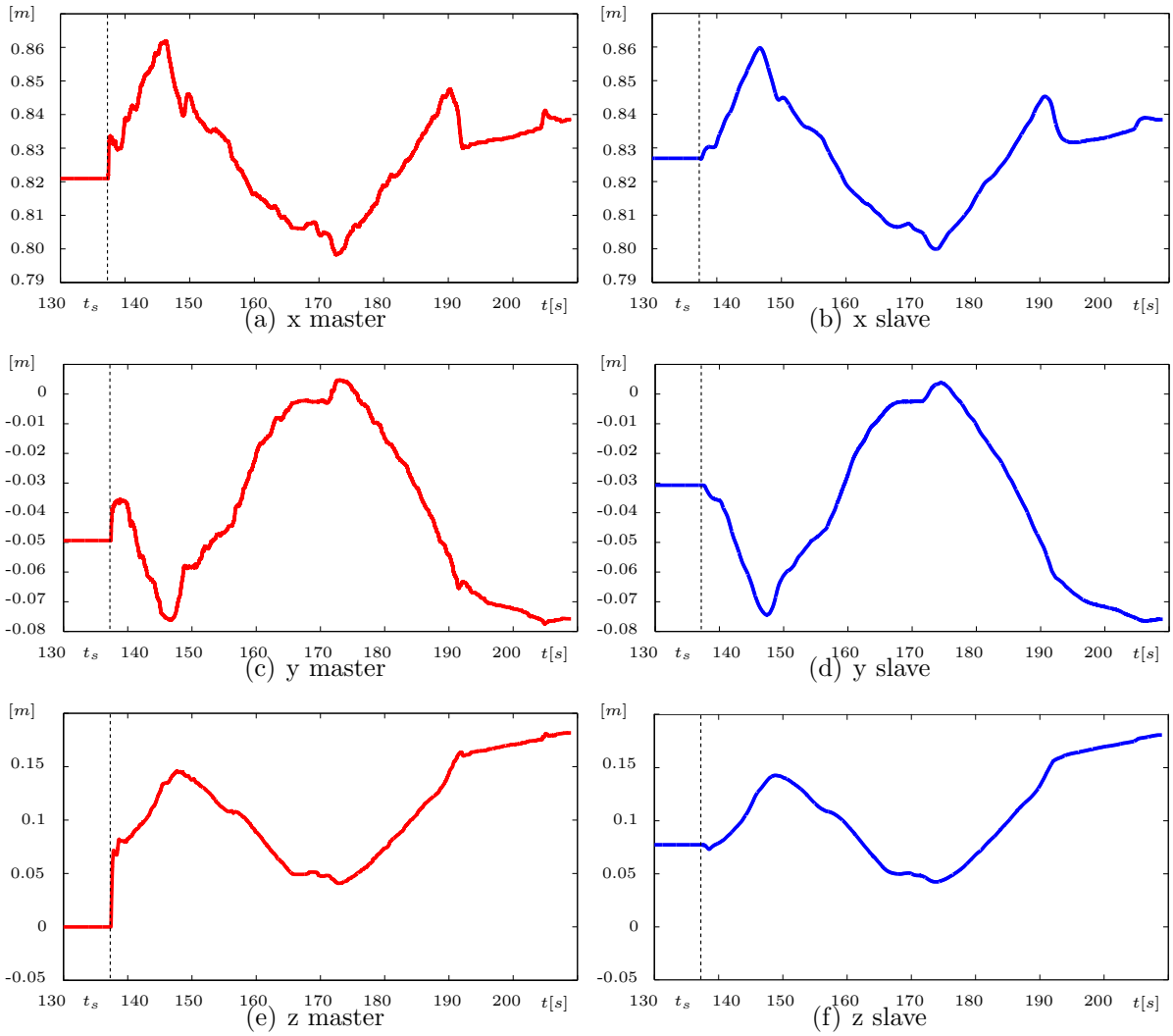


Figure 4.21: Cartesian positions of master and slave during the teleoperation phase. At the switching instant t_s, master and slave are not aligned while, after a short transient, the coupling between the two motions becomes effective.

Figure 4.23 shows the energy level of the tanks at the master and slave sides. It can be seen that the master stores energy until it reaches the upper bound $\bar{T}$ around $t = 154$ s. The energy not stored in the tank is then sent to the slave tank, which increases its level faster than before.

This test shows that the exchange of energy during the alignment and the normal

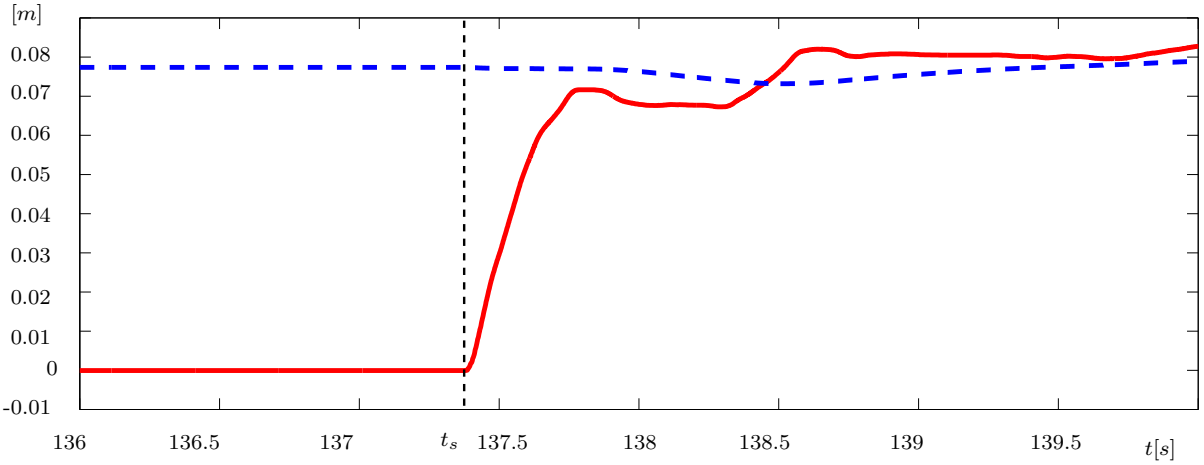


Figure 4.22: Detail of the z coordinate of master (red solid line) and slave (blue dashed line) cartesian position during the transient. At the switching instant t_s, master and slave are not aligned. Then, the master starts moving towards the slave position and, at the end of the alignment phase, the teleoperation is completely operative and master and slave are fully coupled.

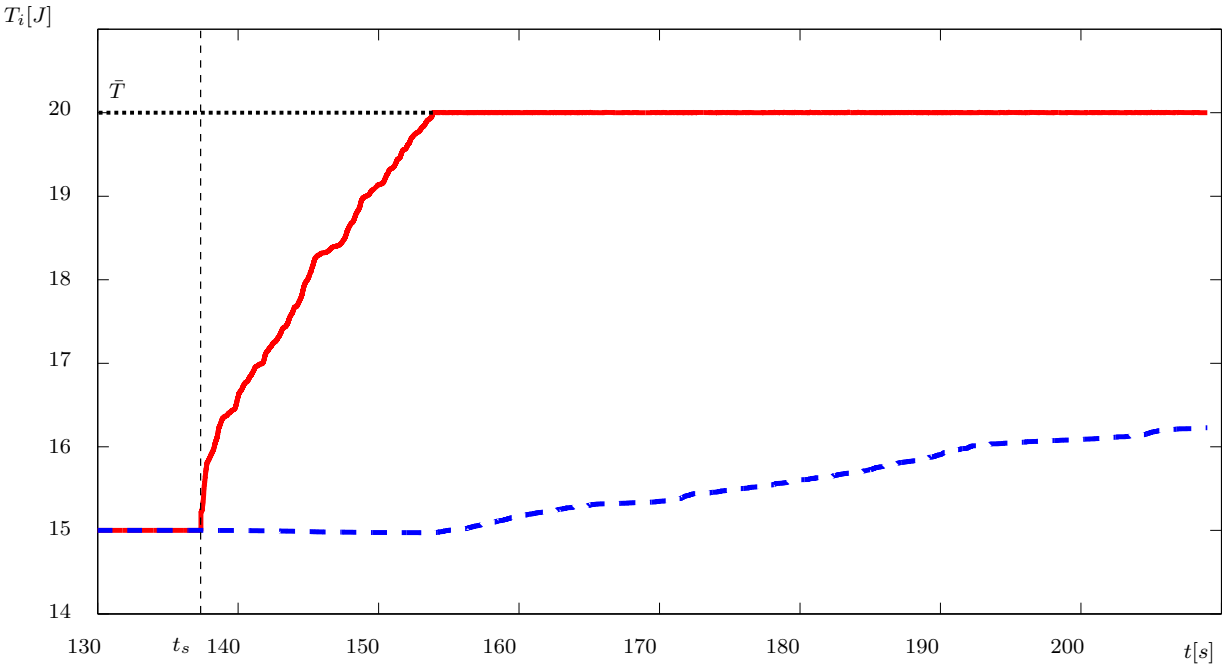


Figure 4.23: Energy level of the tanks at the master (red solid line) and slave (blue dashed line) sides when the teleoperation mode is active.

teleoperation works as expected.

It is worth noting that, during the transition from the "locked" slave robot to the teleoperated slave robot, there are no spikes, oscillations or abrupt movements. The proposed control architecture allows to safely switch from fully autonomous mode to teleoperated mode, which is crucial for risky and delicate applications such as robotic surgery.

Validation of the variable impedance control on a different setup

As a proof of concept to validate the variable impedance control strategy, further experiments on a different setup including a different robot have been performed. The simplified setup (Fig. 4.24) reproduces the puncturing scenario and, in particular, the use case of the cryoablation procedure for kidney tumor treatment.

The IceRod cryoablation needle has been mounted on the end-effector of a Barrett WAM 7–DOF [95]. The algorithm is implemented in C++ and interfaced to the WAM through a software library provided by Barrett.

The object to be punctured (i.e. the *phantom*) has not the shape of a human organ but it is simply a three-layers box. The layers are made of gel wax, having different percentage of paraffin inclusion and, therefore, different stiffness. The upper layer has a medium stiffness and is meant to represent the human skin, the intermediate layer represents the kidney and has the lowest stiffness, while the last one stands for the tumor and is the most rigid layer.

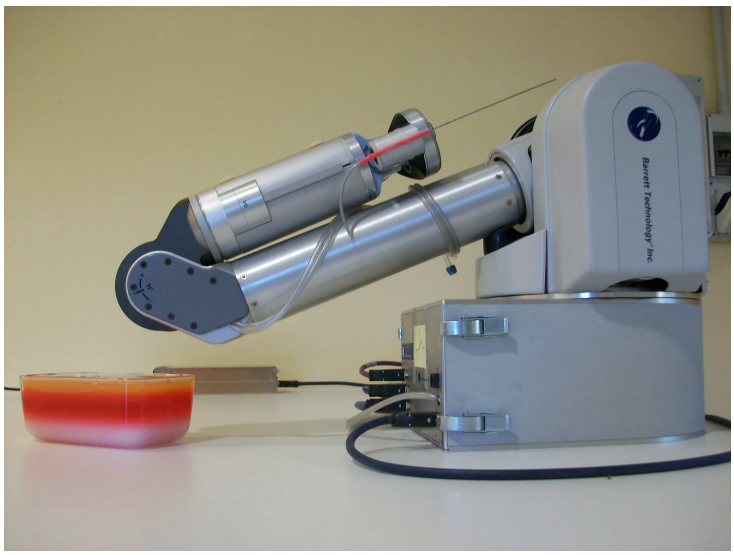


Figure 4.24: Simplified experimental setup reproducing the puncturing scenario.

The end-effector of the robot is moved to an initial Cartesian configuration $P_{start} =$

$[-0.313\ -0.242\ 0.025]^T$ and has to insert the needle from a point $P_0 = [-0.25\ -0.20\ 0.05]^T$, in which the needle is still out of the phantom, to a point $P_1 = [-0.35\ \ -0.11\ \ -0.14]^T$ that brings the needle tip into the lowest layer (i.e. inside the tumor). The phantom is located along the z-direction between P_0 and P_1.

The robot end-effector has to reach each point following a linear trajectory with a fixed orientation. Since the end-effector orientation never changes, in the following will be reported only the results regarding the translational coordinates x, y and z. Therefore, the end-effector has to

1. move from the homing position to P_{start} using a standard position controller,
2. move from P_{start} to the initial point P_0 using the impedance controller,
3. move from P_0 to the target point P_1, using the impedance controller and making the needle cross the phantom layers with the given stiffness, different for each layer,
4. move back from P_1 to P_0, again using the impedance controller.

In order to reach the steady state, when the end-effector arrives in P_0 or in P_1, it stops for 10 seconds before moving forward. Figure 4.25 shows the end-effector Cartesian position: the dash-dot lines represent the reference trajectories, while the solid lines represent the end-effector actual motion. In Fig. 4.25 and in the following, *needle insertion* stands for the end-effector movement from P_0 to P_1 and includes either the needle approach to the phantom in free motion or the layers crossing, while *needle extraction* is considered to be the opposite movement, i.e. from P_1 to P_0.

The desired damping and stiffness matrices are chosen as diagonal matrices while the desired inertia matrix is chosen equal to $\Lambda(x)$, where $\Lambda(x)$ is computed as described in Sec. 2.3.2. Thus, the impedance model used to control the robot is the one given in (2.36).

For the damping, the diagonal elements are chosen as 1 Ns/m, while the entries for the stiffness matrix change depending on the layer that has to be crossed by the needle. In the specific, Table 4.1 presents the diagonal elements for the translational part of the matrix $K_d(t)$ adopted in the variable impedance control, while Fig. 4.26 shows the stiffness evolution over time. The transition from one stiffness level to another takes place by means of a ramp with high slope, as the time that elapses to cross each layer is short.

The stiffness values are experimentally determined. Following the approach described in [35], experiments to derive accurate profiles of the stiffnesses exerted during the puncturing could be performed by means of a data-driven identification procedure.

As expected, Fig. 4.25 shows that the system remains stable despite the several changes of stiffness. In fact, the end-effector actual motion follows the reference trajectory and does not diverge from it over time.

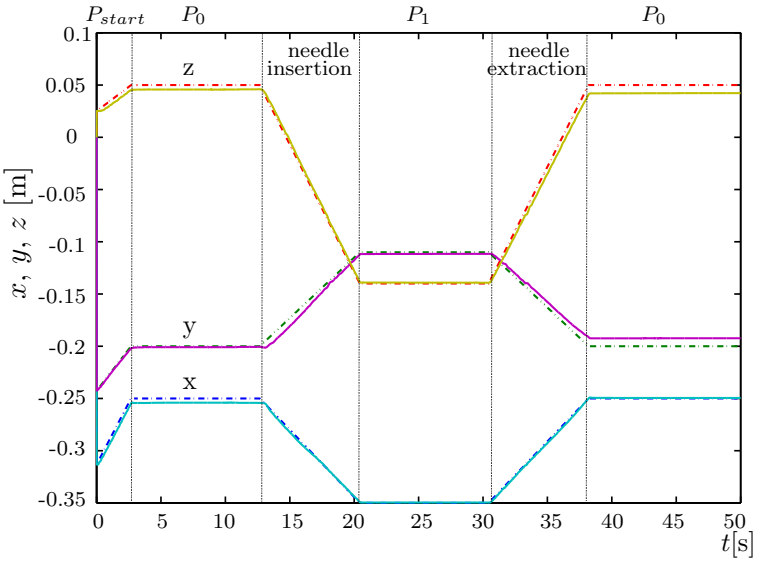


Figure 4.25: Desired trajectory (dash-dot lines) and effective end-effector position (solid lines) during the puncturing experiment.

Table 4.1: Desired stiffness: $K_d(t) = K_c + K_v(t)$

Layer	$\mathbf{K}_c$ **[N/m]**	$\mathbf{K}_v$**(t) [N/m]**	**Insertion [sec]**	**Extraction [sec]**
out of phantom	1800	0	[0.0; 17.70]	[33.26; 50.0]
upper *skin*	1800	700	[17.71; 18.45]	[32.67; 33.25]
intermediate *kidney*	1800	400	[18.46; 19.50]	[31.60; 32.66]
lower *tumor*	1800	1000	[19.51; 25.54]	[25.55; 31.59]

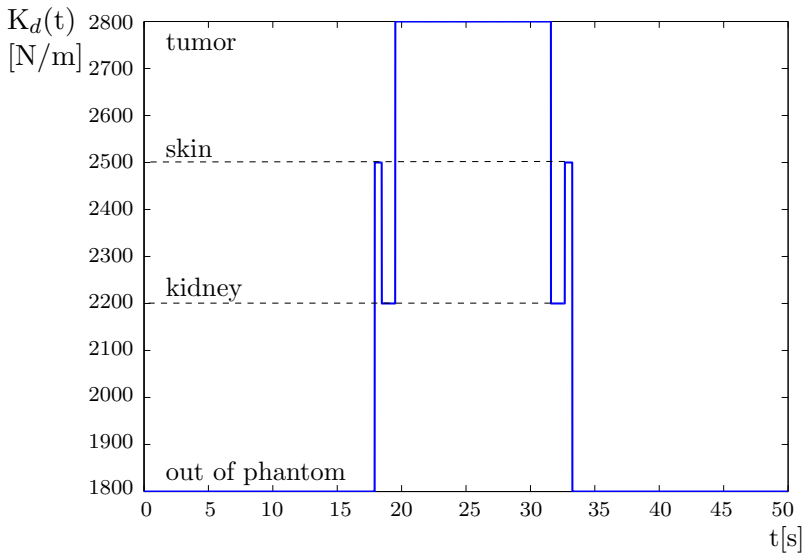


Figure 4.26: Evolution over time of the values chosen as diagonal elements for the desired stiffness matrix $K_d(t)$ during needle insertion and extraction.

Figure 4.27 shows the behavior of the tank energy T. When the needle insertion starts ($t = 13$ sec), since the end-effector moves in free motion, the tank begins to store the dissipated energy. When the needle reaches the first layer, the stiffness changes and the required energy to implement this behavior is extracted from the tank. Then, during the layer crossing, the stiffness remains constant and, consequently, the tank does not need to be exploited and the energy raises again. The same behavior can be noticed when the needle crosses the following two layers: first the energy falls, due to stiffness changing and consequent tank exploitation, then it raises again and remains constant until the following change of stiffness is required (i.e. the following layer has been reached). During the steady state between insertion and extraction, the tank energy remains constant. Then, during the extraction, since this action is passive and not active as the previous one, in order to implement the changes of stiffness, energy is released by the system and consequently the energy stored in the tank raises. When the needle exits the phantom, the tank energy increases, until the end-effector reaches again the initial point P_0.

Little energy has been extracted or inserted into the tank because, as shown in Fig. 4.25, the trajectory error $\tilde{x}(t)$ is small and then, from (2.45), the values of $w(t)$ used in (2.43) are small too.

4.4 Suturing

Figure 4.28 shows the robotic platform for the autonomous suturing. Two micro units are mounted on the macro structure of the I-SUR robot. The right arm of the robot

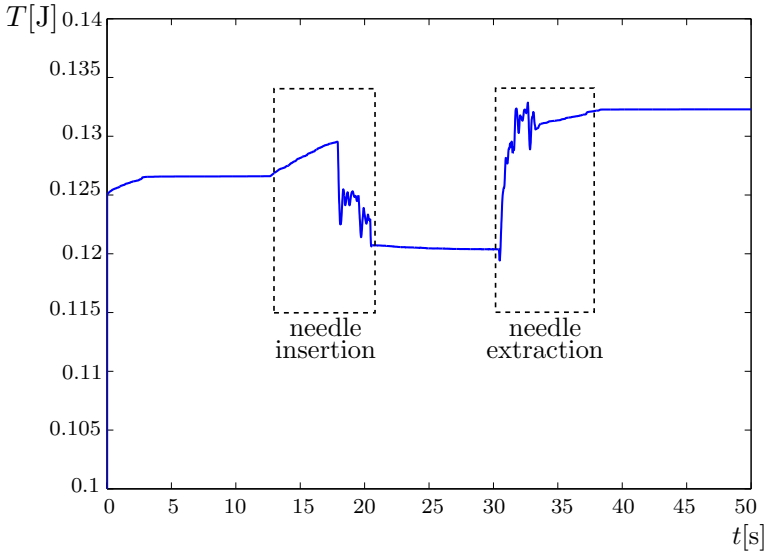


Figure 4.27: Behavior of the tank energy T during needle insertion and extraction.

holds the Covidien Endo Stitch and provide 4 DOFs, while the left arm holds the tool to remove the suturing thread and provides 6 DOFs. The surgeon can require to switch to teleoperated mode in every moment by simply selecting the button on the surgical interface. Two PHANTOM Omni haptic devices are included into the setup to provide the teleoperation functionality on both the arms.

For the suturing tests, a suturing pad with similar mechanical properties as human abdomen cutaneous and subcutaneous layers has been used (Fig. 4.29). A planar wound has been obtained by simply slicing the pad.

4.4.1 Planning of the surgical task

As for the puncturing task, the first step when automating the suturing procedure is the planning of the surgical task. The Bumblebee2 stereo camera captures the right and left images of the suturing pad and the sensing system, by exploiting image processing techniques, recognizes the edges of the wound.

The goal of the planning tool is to compute the correct placement of the stitching points, while satisfying the following suggestions and guidelines obtained from interviews to expert vascular surgeons.

- The first stitch should start at the beginning of the wound
- The distance from the wound edge and the stitching point should be 5 mm
- The distance between the stitches should be 5 mm

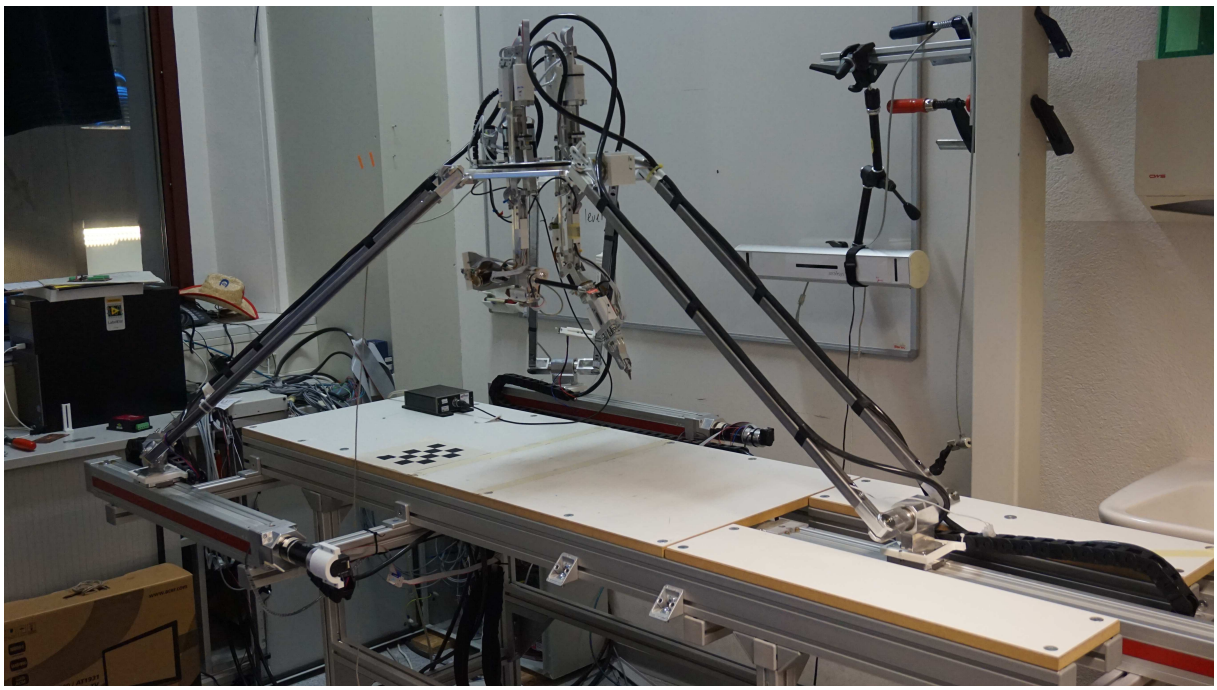


Figure 4.28: The overall robotic platform for suturing. The right arm holds the Covidien Endo Stitch while the left arm holds the gripper.

Figure 4.29: Suturing pad reproducing the cutaneous (yellow) and subcutaneous (red) layers of the human abdomen.

The I-SUR suturing planner (Fig. 4.30) loads the detected edges from the sensing system and the user settings, giving as output a feasible planning of the stitching points. As shown in Fig. 4.31, first the main axis of the wound is detected. Then, being the wound a straight line, the stitching points are placed 5 *mm* away from the edges, as required, lying on the perpendicular lines to the main axis (Fig. 4.32). The algorithm can run after each stitch to adapt the planned points while the gap becomes more and more narrow. For more irregular wounds, the main axis may be represented as a more complex line fitting the shape of the wound.

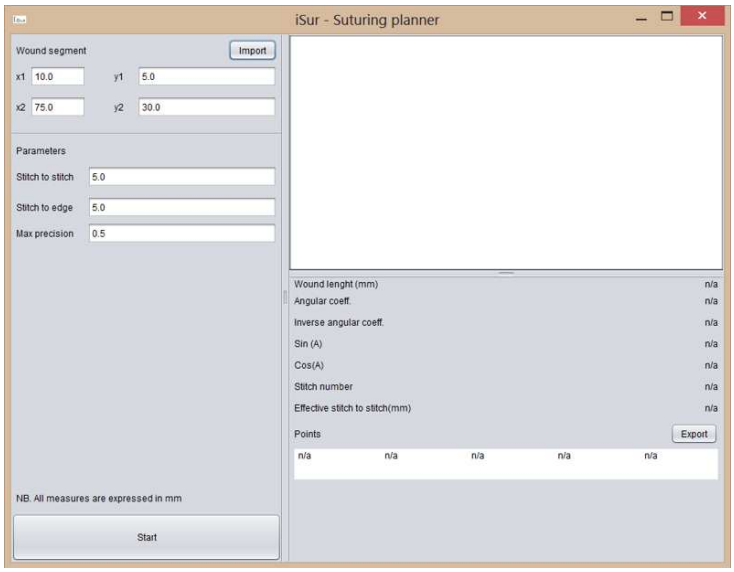


Figure 4.30: View of the I-SUR suturing planner.

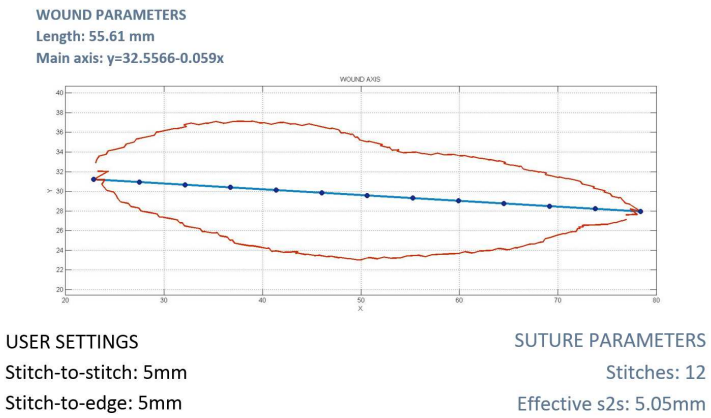


Figure 4.31: The algorithm detects the main axis of the wound, given the user settings.

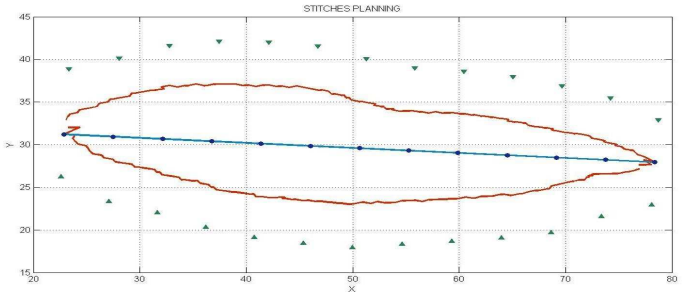


Figure 4.32: The algorithm computes the planning of the stitching points. The stitches lying on the perpendicular lines to the main axes and are placed 5 *mm* away from the edges.

4.4.2 Software architecture

The task-related state machine (i.e. the Task Supervisor state machine) can be developed to include different operating scenarios other than the puncturing task as, for example, the autonomous execution of a wound suturing. A simplified state machine including only the macro-states of the suturing task is shown in Fig. 4.33. The procedure starts when the right end-effector, i.e. the one holding the Endo Stitch, is in a pose away from the wound. The thread has been pulled by the left end-effector and it is in tension. Then, the right end-effector moves to a reference pose over the wound, with the orientation aligned to the wound direction, and starts to move towards the wound. When it reaches the target pose inside the wound, the pitch of the right end-effector is changed to rotate the Endo Stitch to the right planned stitching point. When the pitch is almost parallel to the skin plane, the stitch is applied. Then, the right end-effector moves away in order to pull the thread, while the left end-effector hook the thread and push it to a position where the tension of the thread is ensured. In this way, half of a complete stitch, i.e. the right stitch, is performed. The procedure is repeated to apply the left stitch and thus the first stitch is completed. The whole procedure is then repeated for all the stitches computed by the suturing planner.

The main features of the software architecture described in Chapter 3 are its flexibility, reconfigurability and modularity. Indeed, the suturing task is a proof of concept that small adaptation to the architecture presented in Section 4.3.2 can allow to easily implement other surgical tasks. The full deployment for the considered case study is shown in Fig. 4.34. The core part of the robotic control system is still implemented using the same Orocos component-based approach described in Section 4.3.2 and runs on PCs with Ubuntu 12.04 Linux operating system and Xenomai hard-real time extension. The complete software setup adopted for experiments integrates some parts not Orocos-based.

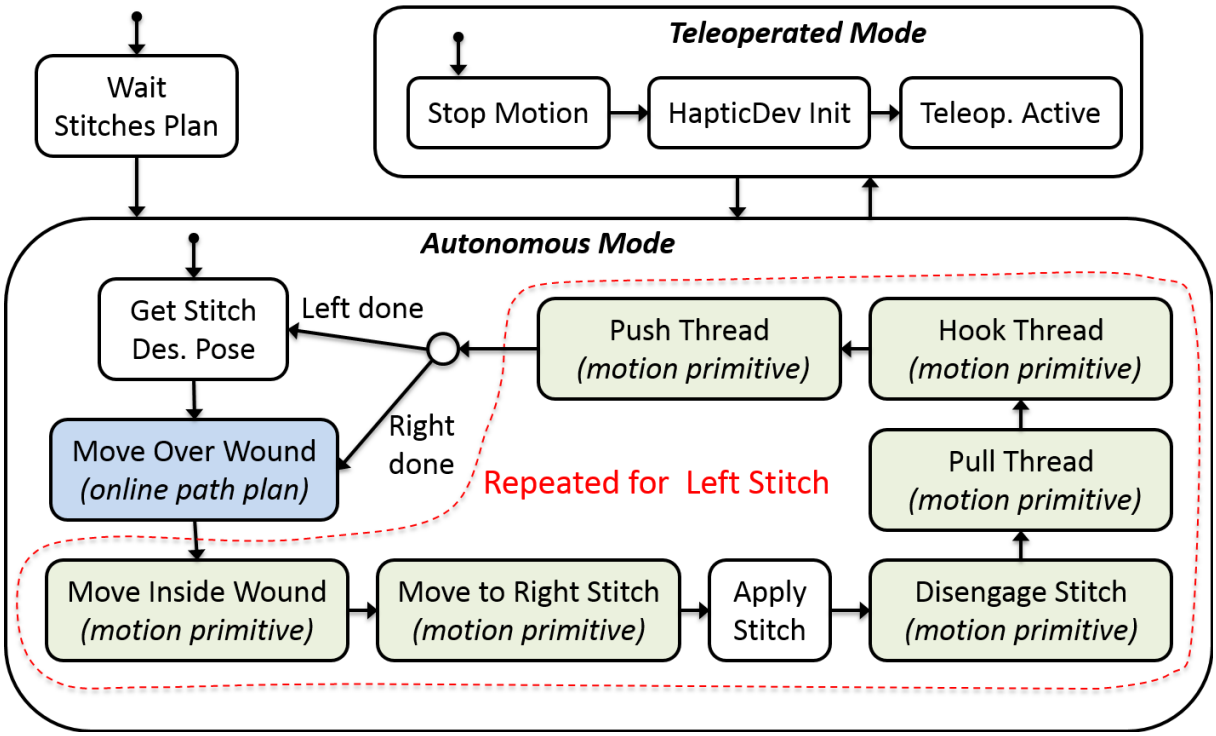


Figure 4.33: Task Supervisor simplified state machine for the suturing task.

The main differences between the architecture for the puncturing task described above and the suturing task are the following:

- Since it represents the formal description and sequence of instruction of the specific task, the state machine for the Task Supervisor is different.
- The setup includes only one robot (i.e. the I-SUR robot) but two micro units are mounted on the same macro structure. This results in a dual-arm manipulator where the two arms are constrained each other. The motion planner has to compute only one path but in the state space SE(3)xSE(3) (left/right arm tips). Collision-free paths for the dual arm robot are generated using the random sampling algorithm RRT-Connect as for the puncturing task.
- The sensing part processes the images coming from a Bumblebee2 stereo camera to detect the wound edges and the tip of the Endo Stitch. The algorithms are developed using the ROS framework. However, ROS nodes and Orocos can easily interoperate as shown in [96] and [97].
- Two PHANTOM Omni are used to teleoperate both the arms of the I-SUR robot. However, this results in simply creating two instances of the same *Haptic Inter-*

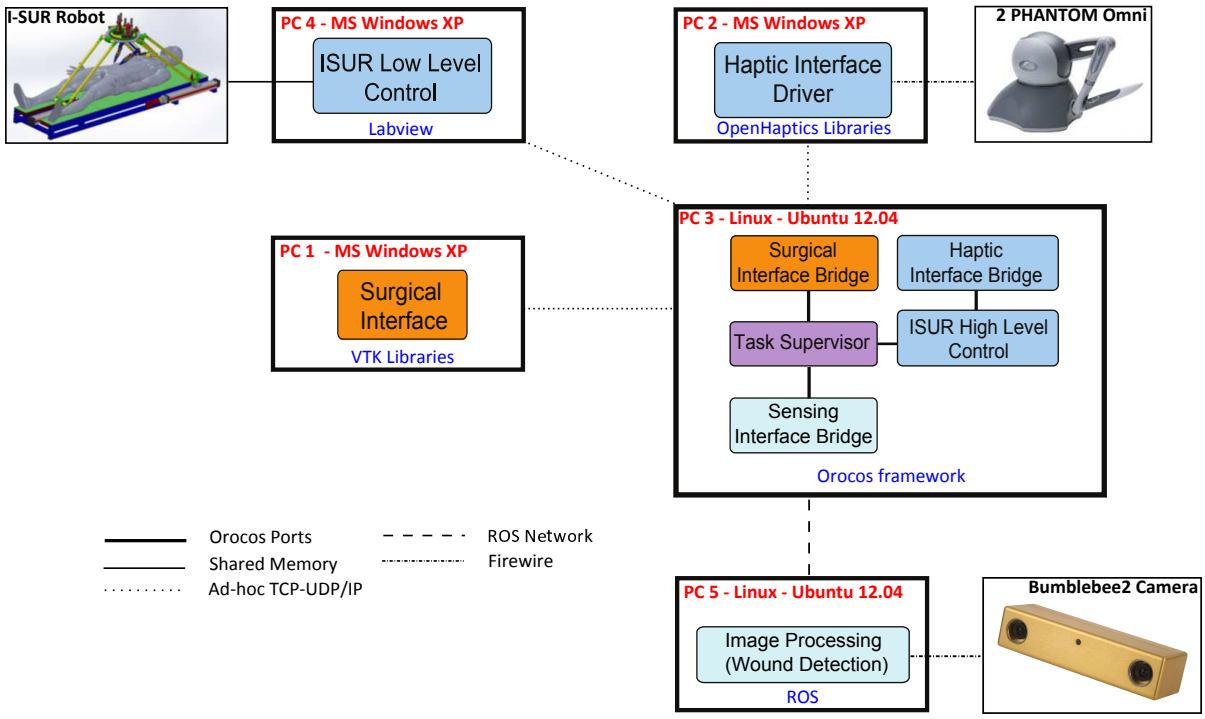


Figure 4.34: Distributed deployment of the component-based software architecture for autonomous surgical robots performing the suturing task.

face Bridge Orocos component, one for the right PHANTOM and one for the left PHANTOM.

As it can be easily noticed, all these differences require only small software adaptation to the overall architecture.

4.4.3 Experimental validation of the control architecture

Since several tests validating the control strategies presented in Chapter 2 have been already presented related to the puncturing task, the focus of this section will be the control architecture.

A high-fidelity 3D viewer reproducing the I-SUR robot (Fig. 4.35) has been developed to ease and speed up the validation phase. As shown in Fig. 4.36, the software architecture behind the *I-SUR Robot Visualizer* is exactly the same of the real setup (Fig. 4.34). In this way, the full architecture can be tested and validated by running all the software components with the I-SUR Robot Visualizer instead of the real robot interface. The simulated setup allowed an accurate analysis of task feasibility, thanks to the reconfigurability of the motion primitives and online collision-checking, and allowed to perform tests even when the real robotic setup was not fully operational for several reasons (e.g.

temporary hardware failure or tests, laboratory space occupied for different tests, etc.). In the real setup, the only adaptation that had to be done was the replacement of the I-SUR Robot Visualizer with the I-SUR Robot low-level control interface.

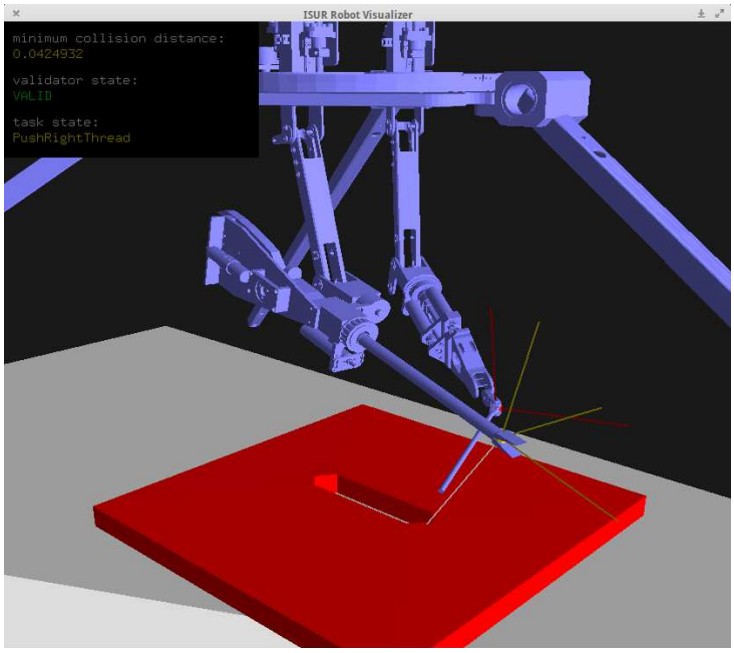


Figure 4.35: 3D viewer reproducing the I-SUR robot. In the top-left corner the task state, the distance checking and the kinematics status are shown.

The 3D viewer provides information about the distance from the nearest obstacle, the status of the kinematics (i.e. valid, collision risk or out of workspace) and the current state of the task state machine (see top-left corner in Fig. 4.35).

The I-SUR Robot Visualizer has been used for performing the suturing task and for teleoperation both the arms of the robot when the intervention of the user is required (Fig. 4.37). Then, the same procedure has been performed on the real setup, by simply switching off the viewer and connecting the driver of the I-SUR robot.

The teleoperation of the dual arms system has been performed using the algorithms presented in Section 2.5. The experimental setup is the one shown in Fig. 4.37. Two PHANTOM Omni haptic devices were utilized as master devices of the teleoperation system, while the two slave arms are the end-effector of the I-SUR Robot. The validating tests have been performed using the I-SUR Robot Visualizer.

The user moves the two haptic devices and is guided along the reference paths (black lines in Fig. 4.38), feeling the influence of the attractive potential field. Then, he decides to move away from the paths without considering the effect of the attractive forces. The effect of these forces, is clearly shown in the case of the end-effector B in Fig. 4.38,

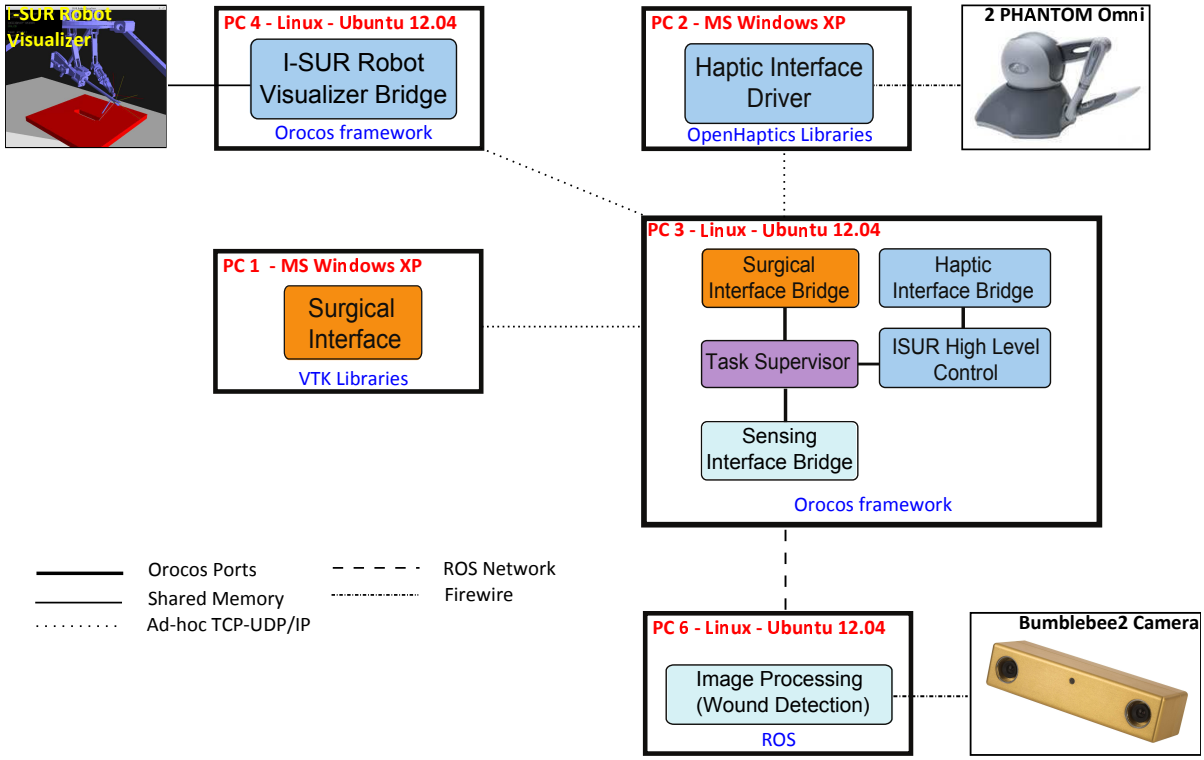


Figure 4.36: Distributed deployment of the component-based software architecture for autonomous surgical robots performing the suturing task. The I-SUR robot is replaced by the I-SUR Robot Visualizer.

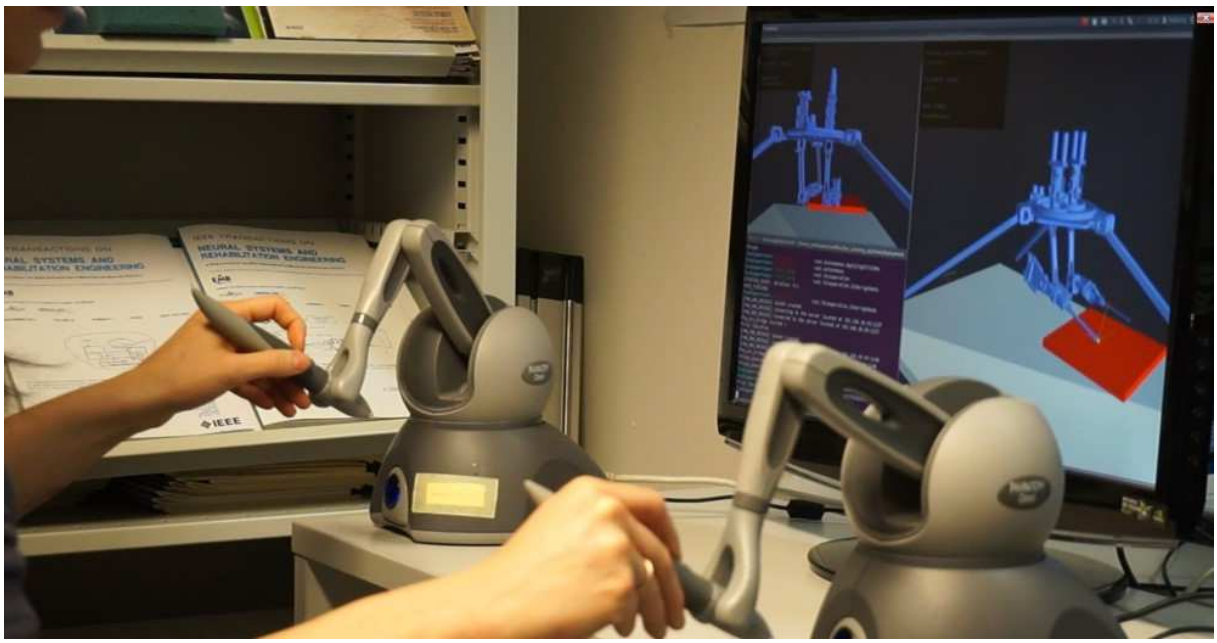


Figure 4.37: Teleoperation of the two arms of the I-SUR robot.

where the blue arrows correspond to the rotated assistive forces computed as described in Section 2.5 and the red lines show the motion of the master devices. Since the end-effector B doesn't touch any obstacle, the only effect that the user feels is the attractive potential field that tends to bring the master B back on the reference path.

Conversely, while moving freely, the end-effector of the master device A touches the contour of the spherical bounding box around an obstacle. A detailed view of the effect of the rotated assistive force when the robot feels the effect of the repulsive potential field is shown in Fig. 4.39. The black dotted line represents the path of the master device A, while the solid and thick black segment represents the current set-point $\hat{t}_A$ where the rotated assistive force (blue arrows) is directed according to the considerations presented in Section 2.5. In this zoomed view it is possible to notice how the standard direction of the virtual force (magenta arrows) would be less efficient than the rotated assistive force.

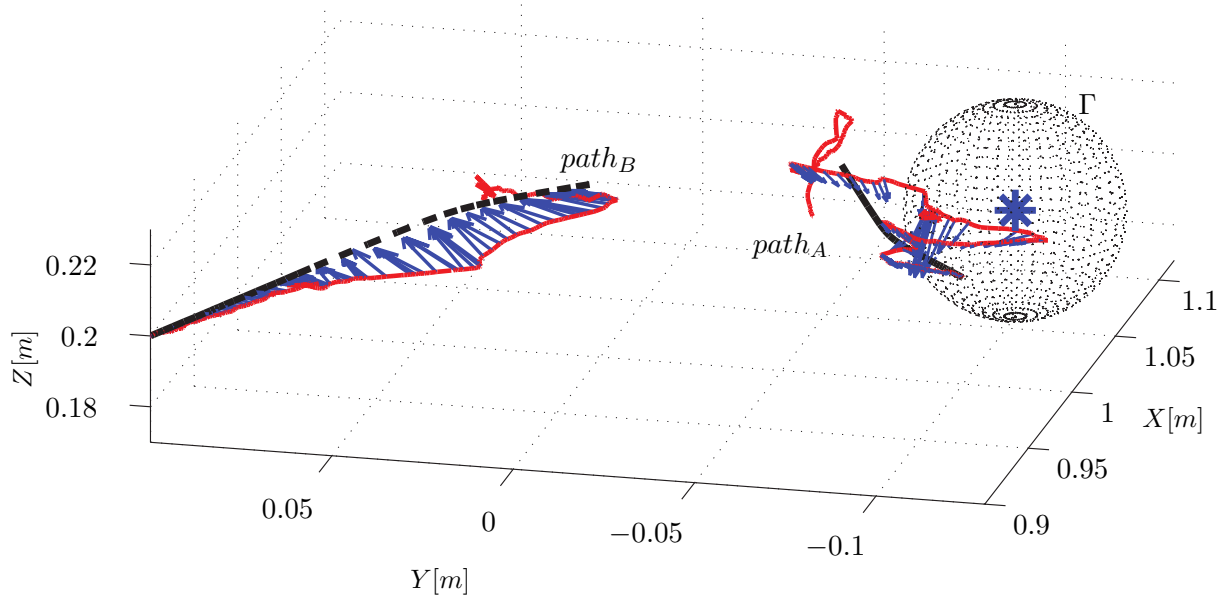


Figure 4.38: Visualization of the two reference paths (black lines), of the positions of the master end-effectors (red lines) and of the rotated assistive forces applied to the master devices (blue vectors).

Figure 4.40 and Fig. 4.41 show the rotated assistive forces and the assistive forces without the rotation reflected to the two master devices. In particular, Fig. 4.40 reports the assistive forces felt by the master A during the time interval in which its position is within the path shown in Fig. 4.39. The assistive forces are effectively rotated to guide the master towards the reference path and the difference respect the standard forces is highlighted. Instead, Fig. 4.41 shows that the rotated forces correspond to the standard forces, since the master B feels only the effect of the attractive forces.

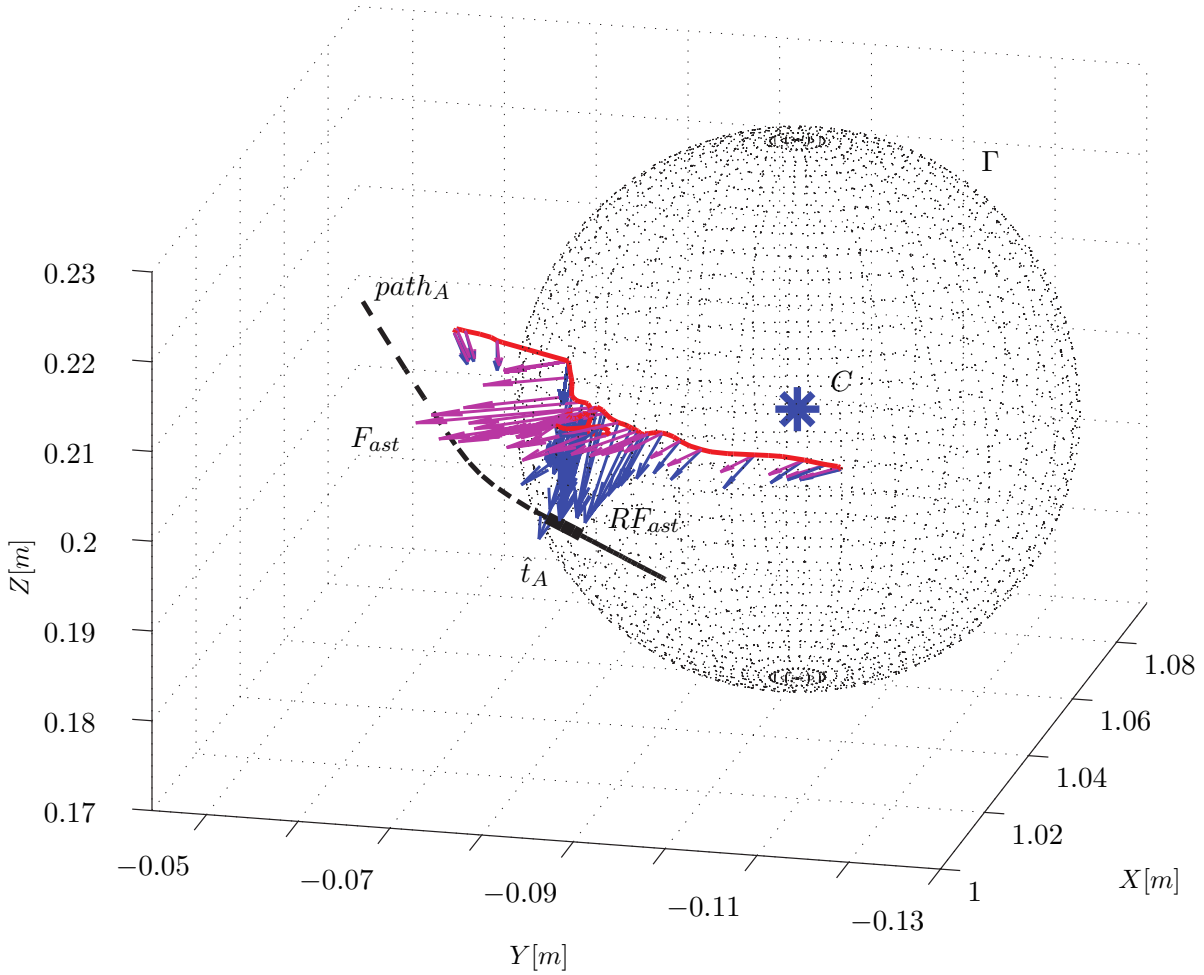


Figure 4.39: Zoomed view of the part of the path where the end-effector of master A starts feeling the effect of the repulsive potential field. The rotated assistive forces (blue arrows) guide the master towards the current set-point $\hat{t}_A$, while standard fixture forces (magenta arrows) would move the robot away from the desired set-point.

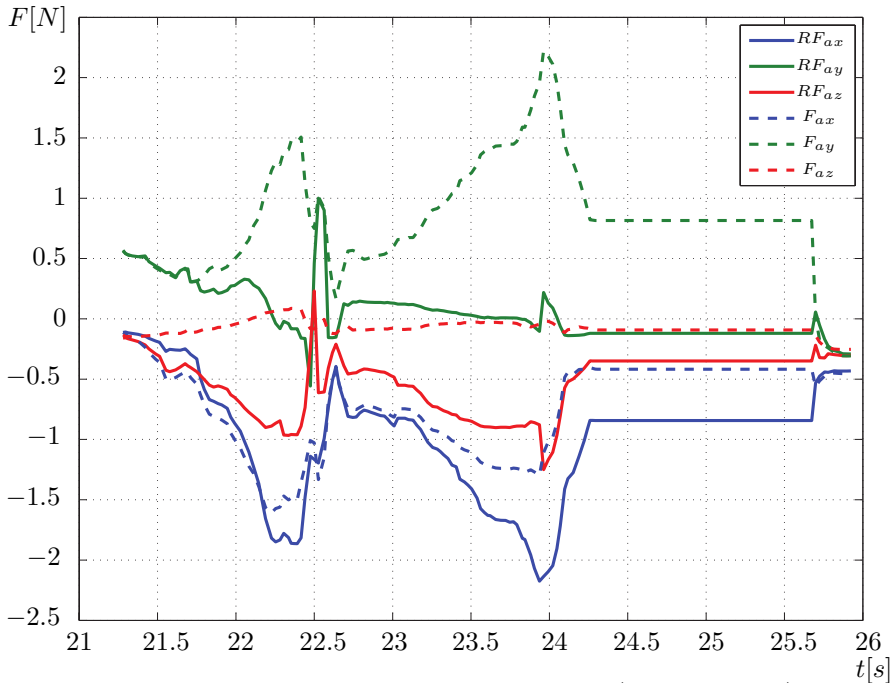


Figure 4.40: Evolution over time of the assistive forces (dotted lines) and of the rotated assistive forces (solid lines) for the master device A when the user feels the effect of the repulsive potential field.

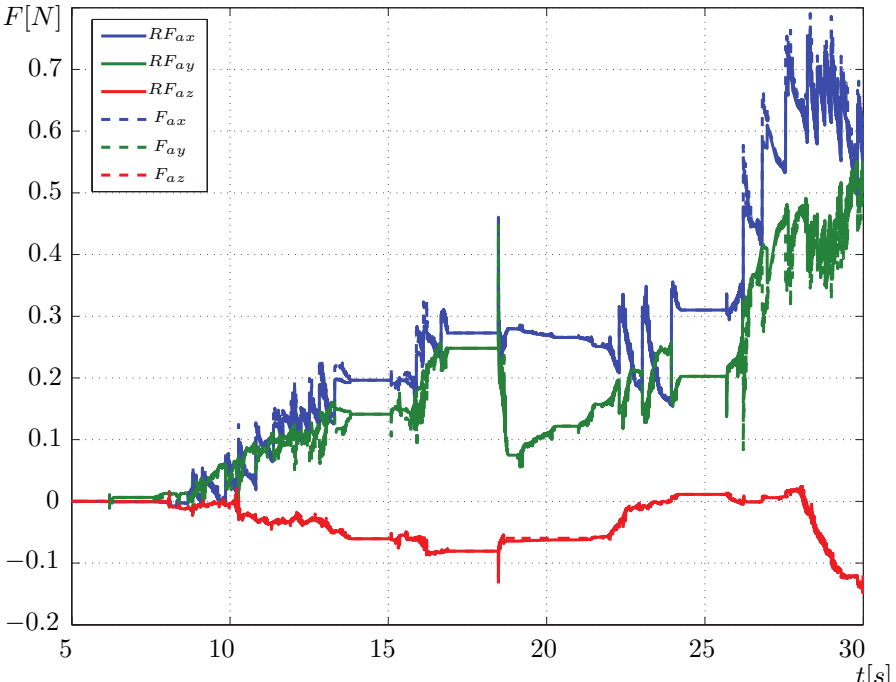


Figure 4.41: Evolution over time of the assistive forces (dotted lines) and of the rotated assistive forces (solid lines) for the master device B. Since the robot doesn't move near obstacles, the only effect felt by the user is the attractive potential field and thus the rotated forces correspond to the original assistive forces.

4.5 Discussion

Validation of theoretical findings is a critical task when developing new strategies and methodologies. Providing proofs of concept and possible applications of the theoretical parts allow to understand practically the advantages and the usefulness of what has been proposed. The control strategies and the software architecture presented in Chapters 2 and 3 have been implemented on the semi-autonomous robotic surgical system designed within the I-SUR project and the results of related experiments have been discussed in this chapter.

Two particular use cases have been addressed, namely the puncturing task and its application to cryoablation of kidney tumors and the suturing of planar wounds.

The first use case has been discussed more intensively. A tool for automatically planning the number and the correct placement of the cryoprobes in cryoablation (i.e. the Cryo-planner) has been developed and provides the input to start the procedure. The results of the implementation of the variable admittance/impedance control and of the two-layer bilateral teleoperation architecture have been presented and discussed in order to validate the theoretical results presented in this thesis. Specific experimental scenarios have been prepared to provide a proof of concept of how it would be possible to automate simple surgical tasks.

The second use case, the suturing task, allowed to demonstrate the reconfigurability and flexibility features of the software architecture. In fact, only small adaptation to the setup and to the control architecture allowed to completely automate the suturing procedure, again from the planning of the surgical task to the automatic execution of the operation. Moreover, the teleoperation of the dual arms robotic platform with guidance virtual fixtures has been implemented and tested.

Chapter 5

Concluding remarks

Autonomous surgical systems are still in a very preliminary conceptual phase. As they progress, they will be useful in the operating room to allow the surgeon to focus only on the most critical parts of the surgical procedure. In this thesis, only the initial steps of the automation process have been addressed, trying to focus on handling the safety of the patient. The control strategies and the software architecture proposed in this thesis focused on stability and passivity of the system in order to guarantee safety requirements needed when dealing with robotic systems.

Since the robot has to interact with environments mostly unknown and usually deformable, the control of the interaction behavior of the robot is fundamental. Furthermore, it might happen that the autonomous system cannot complete the procedure due to unexpected events. In this case, the intervention of the surgeon is mandatory and the transition from autonomous mode to teleoperated mode has to be carefully managed to avoid abrupt movements and oscillations that could harm the patient.

This thesis dealt with the development of control strategies that allow a safe interaction between a robot and the environment in critical tasks, such as automation of simple surgical operations. In particular, a passivity-based interactive control architecture that allows to implement safe and stable time-varying interactive behaviors has been presented. The results obtained for the variable admittance control have been extended and adapted for implementing a variable impedance control. To address the problem of safely switch to teleoperation, a two-layered bilateral control architecture has been described. Then, a strategy to teleoperate a dual arms systems being guided with virtual fixtures has been presented.

The proposed automation system has been implemented into a component-based software architecture, characterized by modularity, flexibility and reconfigurability features to address with small adaptations more surgical tasks.

The solutions presented in this thesis have been validated in the realistic surgical

scenario developed within the EU-funded I-SUR project, using a surgical robot prototype specifically designed for the autonomous execution of surgical tasks, e.g. insertion of needles into the human abdomen to perform a kidney cryoablation and the suturing of a planar wound.

Bibliography

[1] Anthony R. Lanfranco, Andres E. Castellanos, Jaydev P. Desai, and William C. Meyers. **Robotic surgery: A current perspective.** *Annals of Surgery*, **239**(1):14–21, Jan 2004.

[2] Y. S. Kwoh, J. Hou, E. A. Jonckheere, and S. Hayati. **A robot with improved absolute positioning accuracy for CT guided stereotactic brain surgery.** *IEEE Transactions on Biomedical Engineering*, **35**(2):153–160, Feb 1988.

[3] B. Davies. **A review of robotics in surgery.** *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, **214**:129–140, Jan 2000.

[4] S. J. Harris, F. Arambula-Cosio, Q. Mei, R. D. Hibberd, B. L. Davies, J. E. Wickham, M. S. Nathan, and B. Kundu. **The Probot – An active robot for prostate resection.** *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, **211**(4):317–325, 1997.

[5] J. Pransky. **ROBODOC – Surgical robot success story.** *Industrial Robot: An International Journal*, **24**(3):231–233, 1997.

[6] U.S. Food and Drug Administration. **Computer-assisted (robotic) surgical systems.** http://www.fda.gov. Accessed date 16/02/2015.

[7] Intuitive Surgical. **Da Vinci surgical system.** http://www.intuitivesurgical.com. Accessed date 16/02/2015.

[8] G. P. Moustris, S. C. Hiridis, K. M. Deliparaschos, and K. M. Konstantinidis. **Evolution of autonomous and semi-autonomous robotic surgical systems: A review of the literature.** *International Journal of Medical Robotics and Computer Assisted Surgery*, **7**(4):375–392, 2011.

[9] Amy Braverman Puma. **Hands-off surgery.** University of Chicago Magazine, Nov–Dec 2008.

[10] Frederick O. Stephens and John W. Niesche. **The use of an sutomatic stapling device for closure of surgical wounds.** *Australian and New Zealand Journal of Surgery*, **44**(4):398–402, 1974.

[11] C. Pappone, F. Vicedomini, G. Manguso, F. Gugliotta, P. Mazzone, S. Gulletta, N. Sora, S. Sala, A. Marzi, G. Augello, L. Livolsi, A. Santagostino, and V. Santinelli. **Robotic magnetic navigation for atrial fibrillation ablation**. *Journal of the American College of Cardiology*, **47**(7), 2006.

[12] H. Mayer, I. Nagy, D. Burschka, A. Knoll, E.U. Braun, R. Lange, and R. Bauernschmitt. **Automation of manual tasks for minimally invasive surgery**. In *Proceedings of the Fourth International Conference on Autonomic and Autonomous Systems*, pages 260–265, March 2008.

[13] N. Padoy and G. D. Hager. **3D thread tracking for robotic assistance in telesurgery**. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2102–2107, September 2011.

[14] P. Fiorini, R. Muradore, G. Akgun, D. E. Barkana, M. Bonfè, F. Boriero, G. De Rossi, R. Dodi, O. J. Elle, F. Ferraguti, L. Gasperotti, R. Gassert, K. Mathiassen, D. Handini, O. Lambercy, L. Li, M. Kruusma, A. Manurung, G. Meruzzi, P. Nguyen, N. Preda, A. Ristolainen, A. Sanna, C. Secchi, and A. E. Yantac. **Development of a cognitive robotic system for simple surgical tasks.** *International Journal of Advanced Robotic Systems*, 2015.

[15] M. Bonfè, N. Preda, C. Secchi, F. Ferraguti, R. Muradore, L. Repele, G. Lorenzi, L. Gasperotti, and P. Fiorini. **Automated surgical task execution: the needle insertion case.** In *Proceedings of the 3rd Joint Workshop on New Technologies for Computer/Robot Assisted Surgery*, Verona, Italy, September 2013.

[16] S. K. Williams, J. de la Rosette, J. Landman, and F. X. Keely. **Cryoablation of Small Renal Tumors.** *EAU-EBU Update Series*, **5**:206–218, 2007.

[17] Adam C. Mues and J. Landman. **Current status of ablative therapies for renal tumors.** *Indian Journal of Urology*, **25**(4):499–507, 2009.

[18] B. G. Patel, C. L. Parsons, M. Bidair, and J. D. Schmidt. **Cryoablation for carcinoma of the prostate.** *Journal of Surgical Oncology*, **63**(4):256–264, 1996.

[19] UR. **Universal Robots.** http://www.universal-robots.com. Accessed date 21/10/2014.

[20] A. Sharon, N. Hogan, and D.E. Hardt. **The macro/micro manipulator: An improved architecture for robot control.** *Robotics and Computer-Integrated Manufacturing*, **10**(3):209–222, 1993.

[21] R. Öpik, A. Hunt, A. Ristolainen, P. M. Aubin, and M. Kruusmaa. **Development of high fidelity liver and kidney phantom organs for use with robotic surgical systems.** In *Proceedings of the IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 425–430, 2012.

[22] Sensable. **PHANTOM Omni haptic device.** http://www.sensable.com. Accessed date 12/02/2015.

[23] K. Mathiassen, D. Dall'Alba, R. Muradore, P. Fiorini, and O. J. Elle. **Real-Time Biopsy Needle Tip Estimation in 2D Ultrasound Images.** In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013.

[24] Pointgrey. **Pointgrey Bumblebee2.** http://www.ptgrey.com/bumblebee2-firewire-stereo-vision-camera-systems. Accessed date 15/02/2015.

[25] F. Ferraguti, C. Secchi, and C. Fantuzzi. **A tank-based approach to impedance control with variable stiffness.** In *Proceedings of the IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 2013.

[26] M. Franken, S. Stramigioli, S. Misra, C. Secchi, and A. Macchelli. **Bilateral Telemanipulation With Time Delays: A Two-Layer Approach Combining Passivity and Transparency.** *IEEE Transactions on Robotics*, **27**(4):741–756, 2011.

[27] F. Ferraguti, N. Preda, G. De Rossi, M. Bonfè, R. Muradore, P. Fiorini, and C. Secchi. **A two-layer approach for shared control in semi-autonomous robotic surgery.** In *Proceedings of the 14th European Control Conference*, Linz, Austria, July 2015.

[28] F. Ferraguti, N. Preda, A. Manurung, M. Bonfè, O. Lambercy, R. Gassert, R. Muradore, P. Fiorini, and C. Secchi. **A Tank-Based Interactive Control Architecture for Autonomous and Teleoperated Robotic Surgery.** *IEEE Transactions on Robotics*, 2015. Submitted.

[29] F. Ferraguti, N. Preda, M. Bonfè, and C. Secchi. **Bilateral teleoperation of a dual arms surgical robot with passive virtual fixtures generation.** In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, September 2015. Submitted.

[30] M. Bonfè, N. Preda, C. Secchi, F. Ferraguti, R. Muradore, L. Repele, G. Lorenzi, and P. Fiorini. **Distributed control architecture for automated surgical task execution with coordinated robot arms.** In *Proceedings of the 19th World Congress of the International Federation of Automatic Control*, Cape Town, South Africa, August 2014.

[31] N. Preda, F. Ferraguti, M. Bonfè, C. Secchi, R. Muradore, and P. Fiorini. **Distributed Software Architecture for Control and Coordination of Autonomous Surgical Robots.** *IEEE Transactions on Industrial Informatics*, 2015. Submitted.

[32] F. Ferraguti, N. Golinelli, C. Secchi, N. Preda, and M. Bonfè. **A component-based software architecture for control and simulation of robotic manipulators.** In *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation*, Cagliari, Italy, September 2013.

[33] M. Torricelli, F. Ferraguti, and C. Secchi. **An algorithm for planning the number and the pose of the iceballs in cryoablation.** In *Proceedings of the 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Osaka, Japan, July 2013.

[34] R. Dodi, F. Ferraguti, A. Ristolainen, C. Secchi, and A. Sanna. **Planning and simulation of percutaneous cryoablation.** In *Proceedings of the 2nd AASRI Conference on Computational Intelligence and Bioinformatics*, Jeju Island, Korea, December 2013.

[35] M. Bonfè, F. Boriero, R. Dodi, P. Fiorini, A. Morandi, R. Muradore, L. Pasquale, A. Sanna, and C. Secchi. **Towards automated surgical robotics: A requirements engineering approach.** In *Proceedings of the IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 56–61, 2012.

[36] N. Hogan. **Impedance Control: An Approach to Manipulation. I – Theory. II – Implementation. III – Applications.** *ASME Transactions Journal of Dynamic Systems and Measurement Control*, **107**:1–24, 1985.

[37] B. Hannaford. **Stability and performance tradeoffs in bi-lateral telemanipulation**. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1764–1767, Scottsdale, AZ, May 1989.

[38] P. F. Hokayem and M. W. Spong. **Bilateral Teleoperation: An historical survey**. *Automatica*, **42**(12), 2006.

[39] S. A. Bowyer, Davies B. L., and Baena F. R. **Active constraints/virtual fixtures: A survey.** *IEEE Transactions on Robotics*, **30**(1):138–157, 2014.

[40] C. Secchi, S. Stramigioli, and C. Fantuzzi. *Control of Interactive Robotic Interfaces: A Port-Hamiltonian Approach.*, **29** of *Springer Tracts in Advanced Robotics*. Springer-Verlag, 2006.

[41] C. Secchi, S. Stramigioli, and C. Fantuzzi. **Position drift compensation in port-Hamiltonian based telemanipulation.** In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4211–4216, 2006.

[42] A. Franchi, C. Secchi, H. I. Son, Heinrich H. Bülthoff, and P. Robuffo Giordano. **Bilateral teleoperation of groups of mobile robots with time-varying topology.** *IEEE Transactions on Robotics*, **28**(5):1019–1033, 2012.

[43] P. Robuffo Giordano, A. Franchi, C. Secchi, and Heinrich H. Bülthoff. **A passivity-based decentralized strategy for generalized connectivity maintenance.** *International Journal of Robotics Research*, **32**(3):299–323, 2013.

[44] D. Lee and K. Huang. **Passive-Set-Position-Modulation Framework for Interactive Robotic Systems.** *IEEE Transactions on Robotics*, **26**(2):354–369, 2010.

[45] L. Villani and J. De Schutter. **Force control.** In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, chapter 7. Springer Berlin Heidelberg, 2008.

[46] L. Barbé, B. Bayle, M. de Mathelin, and A. Gangi. **Online Robust Model Estimation and Haptic Clues Detection during In Vivo Needle Insertions.** In *Proceedings of 1st IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 341–346, Feb 2006.

[47] C. Ott. *Cartesian impedance control of redundant and flexible-joint robots.*, **49** of *Springer Tracts in Advanced Robotics*. Springer-Verlag, 2008.

[48] E. Nuno, R. Ortega, N. Barabanov, and L. Basanez. **A Globally Stable PD Controller for Bilateral Teleoperators.** *IEEE Transactions on Robotics*, **24**(3):753–758, 2008.

[49] D. Lee and M. W. Spong. **Passive Bilateral Teleoperation With Constant Time Delay.** *IEEE Transactions on Robotics*, **22**(2):269–281, 2006.

[50] C. Secchi, A. Franchi, Heinrich H. Bülthoff, and P. Robuffo Giordano. **Bilateral Teleoperation of a Group of UAVs with Communication Delays and Switching Topology.** In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4307–4314, Saint Paul, Minnesota, USA, May 2012.

[51] C. Ott, R. Mukherjee, and Y. Nakamura. **Unified impedance and admittance control.** In *Proceedings of the IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, May 2010.

[52] P. F. Hokayem and M. W. Spong. **Bilateral teleoperation: An historical survey**. **42**(12):2035–2057, 2006.

[53] J. J. Abbott and A. M. Okamura. **Pseudo-admittance bilateral telemanipulation with guidance virtual fixtures**. *The International Journal of Robotics Research*, **26**(8):865–884, 2007.

[54] L. Sciavicco and B. Siciliano. *Modelling and control of robot manipulators*. Springer Science & Business Media, 2000.

[55] G. T. Heineman and W. T. Council. *Component-Based Software Engineering: Putting the pieces together.* Addison-Wesley, 2001.

[56] OROCOS. **The Orocos project.** http://www.orocos.org. Accessed date 12/02/2015.

[57] M. Klotzbücher and H. Bruyninckx. **Coordinating robotic tasks and systems with rFSM statecharts.** *Journal of Software Engineering for Robotics*, **3**(1):28–56, January 2012.

[58] OMPL. **Open Motion Planning Library.** http://ompl.kavrakilab.org. Accessed date 12/02/2015.

[59] Jia Pan, S. Chitta, and D. Manocha. **FCL: A general purpose library for collision and proximity queries**. In *Proceedings of the IEEE International Conference on Robotics and Auomation*, pages 3859–3866, May 2012.

[60] L. Baresi, L. Pasquale, and P. Spoletini. **Fuzzy goals for requirements-driven adaptation**. In *Proceedings of the International Requirements Engineering Conference*, pages 125–134, 2010.

[61] Alloy. **A language and tool for relational models**. http://alloy.mit.edu. Accessed date 19/02/2015.

[62] A. Pnueli. **The temporal logic of programs.** In *Proceedings of the Annual Symposium on Foundations of Computer Science*, pages 46–57, 1977.

[63] E. Torlak and G. Dennis. **Kodkod for Alloy users**. In *Proceedings of the 1st ACM Alloy Workshop*, April 2008.

[64] Object Management Group. **UML v. 2.2 Superstructure specification.** Document N. formal/2009-02-02, 2009. http://www.omg.org/spec/UML/2.2/.

[65] K. L. McMillan. *Symbolic model checking: An approach to the state explosion problem.* Kluwer Academic Publishers, 1993.

[66] K.L. McMillan. *The SMV language.* Cadence Berkeley Labs., 2001 Addison St., Berkeley, USA, March 1999.

[67] H.Q.P. Nguyen, O.J. Elle, D. Handini, and K. Mathiassen. **Intra-operative Reasoning and Situation Awareness Algorithm for Kidney Tumor Cryoblation by Robot.** In *24th Int. Conf. of the Society for Medical Innovation and Technology (SMIT)*, September 2012.

[68] M. Bonfè, C. Fantuzzi, and C. Secchi. **Verification of behavioral substitutability in object-oriented models for industrial controllers.** In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain, April 2005.

[69] D. Vanthienen, M. Klotzbücher, and H. Bruyninckx. **The 5C-based architectural composition pattern: Lessons learned from re-developing the iTaSC framework for constraint-based robot programming.** *Journal of Software Engineering for Robotics*, **5**(1):17–35, May 2014.

[70] ROS. **Robot Operating System.** http://www.ros.org. Accessed date 21/02/2015.

[71] YARP. **Yet Another Robot Platform.** http://wiki.icub.org/yarp. Accessed date 21/02/2015.

[72] OpenRTM. **An open-source robotic technology middleware.** http://www.openrtm.org. Accessed date 12/02/2015.

[73] Orca. **Components for robotics.** http://orca-robotics.sourceforge.net. Accessed date 12/02/2015.

[74] GenoM. **Generator of modules.** http://www.openrobots.org/wiki/genom. Accessed date 12/02/2015.

[75] Peter Soetens. *A software framework for real-time and distributed robot and machine control.* PhD thesis, Department of Mechanical Engineering, Katholieke Universiteit Leuven, Belgium, May 2006. http://www.mech.kuleuven.be/dept/resources/docs/soetens.pdf.

[76] CORBA. **Common Object Request Broker Architecture) specifications.** http://www.corba.org.

[77] Markus Klotzbücher, Peter Soetens, and Herman Bruyninckx. **OROCOS RTT-Lua: An execution environment for building real-time robotic domain specific languages**. In *Proceedings of the International Workshop on Dynamic Languages (DYROS)*, Darmstadt, Germany, November 2010.

[78] J. Kuffner and S.M. LaValle. **RRT-connect: An efficient approach to single-query path planning**. In *Proceedings of the IEEE International Conference on Robotics and Auomation*, pages 995–1001, 2000.

[79] KDL. **Kinematics and Dynamics Library.** http://www.orocos.org/kdl. Accessed date 13/02/2015.

[80] T. K. Podder, J. Sherman, D. P. Clark, E. M. Messing, D. J. Rubens, J. G. Strang, L. Liao, R. A. Brasacchio, Y. Zhang, W. S. Ng, and Y. Yu. **Evaluation of robotic needle insertion in conjunction with in vivo manual insertion in the operating room.** In *Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication*, pages 66–72, Aug 2005.

[81] B. Maurin, L. Barbe, B. Bayle, P. Zanne, J. Gangloff, De Mathelin M., A. Gangi, L. Soler, and A. Forgione. **In vivo study of forces during needle insertions.** In *Proceedings of the Medical Robotics, Navigation and Visualization Scientific Workshop*, pages 415–422, 2004.

[82] O. Khatib. **Reduced effective inertia in macro-/mini-manipulator systems.** In *Proceedings of the International Symposium on Robotics Research*, pages 279–284, 1991.

[83] O. Khatib. **Inertial properties in robotic manipulation: An object-level framework.** *The Inernational Journal of Robotics Research*, **14**:19–36, February 1995.

[84] M. Bouri and R. Clavel. **The linear delta: Developments and applications**. In *Proceedings of the International Symposium on Robotics and German Conference on Robotics*, pages 1198–1205, Munich, Germany, June 2010.

[85] R. Franks and C. Worley. **Quantitative analysis of cascade control**. *Industrial and Engineering Chemistry.*, **48**(6):1074–1079, June 1956.

[86] Galilmedical. *Cryoablation Needles Brochure*, June 2009.

[87] 3D Slicer. **Software for visualization and medical image computing.** http://www.slicer.org. Accessed date 10/02/2015.

[88] David C Lung, Thomas F Stahovich, and Yoed Rabin. **Computerized planning for multiprobe cryosurgery using a force–field analogy.** *Computer Methods in Biomechanics and Biomedical Engineering*, **7**(2):101–110, 2004.

[89] Daigo Tanaka, Kenji Shimada, and Yoed Rabin. **Two–phase computerized planning of cryosurgery using bubble–packing and force–field analogy.** *Journal of Biomechanical Engineering*, **128**(1):49–58, 2006.

[90] Soji Yamakawa and Kenji Shimada. **High Quality Anisotropic Tetrahedral Mesh Generation Via Ellipsoidal Bubble Packing.** In *9th International Meshing Roundtable*, pages 263–274, 2000.

[91] Michael R. Rossi, D. Tanaka, K. Shimada, and Y. Rabin. **Computerized Planning of Cryosurgery Using Bubble Packing: An Experimental Validation on a Phantom Material.** *International Journal of Heat and Mass Transfer*, **51**(23–24):5671–5678, 2008.

[92] Eugene H. Wissler. **Pennes' 1948 paper revisited.** *Journal of Applied Physiology*, **85**(1):35–41, 1998.

[93] T. Skeie, S. Johannessen, and O. Holmeide. **Timeliness of real-time IP communication in switched industrial Ethernet networks**. *IEEE Transactions on Industrial Informatics*, **2**(1):25–39, February 2006.

[94] J. Vila-Carbo, J. Tur-Masanet, and E. Hernandez-Orallo. **An evaluation of switched Ethernet and Linux traffic control for real-time transmission**. In *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 400–407, September 2008.

[95] Barrett Technology. **WAM Arm.** http://www.barrett.com. Accessed date 21/02/2015.

[96] ROS Documentation. **Running ROS across multiple machines.** http://wiki.ros.org/ROS/Tutorials/MultipleMachines. Accessed date 21/02/2015.

[97] ROS Documentation. **Package documentation for rtt_ros_integration.** http://wiki.ros.org/rtt_ros_integration?distro=groovy. Accessed date 21/02/2015.