University of Modena and Reggio Emilia

Department of Sciences and Methods for Engineering

Doctor of Industrial Innovation Engineering

XXXIV cycle

Advanced Control Strategies for Surgical Robotics

Author:
Marco Minelli

Supervisor:
Cristian Secchi

PhD Coordinator:
Franco Zambonelli

October 2021

Reggio Emilia, Italy, March 2022

To my family

Acknowledgements

This work of thesis is the result of a long journey, which lasted several years, thanks to which I was given the opportunity to grow a lot. For this, I want to thank all the people who made it possible. My deepest thanks go first to my supervisor, Prof. Cristian Secchi, for giving me this opportunity and for all the support and help he has given me during these years. I would like to thank Nicola Piccinelli, Giacomo de Rossi and Fabio Falezza from Altair Lab, and Alessio Sozzi from the Lira Lab, for the numerous evenings spent together on the experimental set-up, with whom, more than colleagues, we have become good friends. Special thanks to Federica Ferraguti, Marcello Bonfé and Riccardo Muradore, with whom I had the opportunity to work, and with whom I was able to carry out a lot of work on this thesis. I would like to warmly thank my family, who have always supported me and who have always believed in me. Without you, this would never have been possible. I would also like to warmly thank my girlfriend, Giulia, who believes in me every day, who has always supported me and with whom I have stared another incredible journey. Finally, I would like to thank all those who have been close to me and who have supported me during this journey. Thanks to all of you.

Abstract

Surgical robotics can be considered an established reality in the medical field. The most striking example is the release of the da Vinci® surgical system by Intuitive Surgical almost 20 years ago. It is estimated that more than 200 000 surgeries were carried out with the da Vinci® in 2012 only, most of which were for hysterectomies and prostate removals. Most robotized surgical operations are performed using teleoperation techniques, where the surgeon sits on a remote console and has complete control of the movements of the robot, without the possibility for the robot to make decisions or perform tasks autonomously. The latter is one of the main topics on which researches in surgical robotics are focused nowadays. Indeed, autonomy can give important advantages, which include increased efficiency and repeatability due to precise robot control, improved execution quality thanks to real-time biosignal feedback and computer-aided guidance, and fewer costs. The important developments that have characterized the last decades in terms of artificial intelligence have pushed even further in this direction. However, it is necessary to take into account that making a robotic surgical system autonomous opens up new important challenges in terms of control. The system must be able, for example, to perceive the surrounding environment to cooperate with other systems while avoiding collisions, understand the environment dynamics to interact with it and to compensate external disturbances, and comply with all the constraints necessary to perform the surgical procedure correctly. Another important problem when making a system autonomous is the collection of data necessary for the training of the system, which are often few and difficult to acquire. The work of this thesis starts to address some of these challenging problems arising during the development of the European-funded project Smart Autonomous Robotic Assistant Surgeon. First, a standard laparoscopic procedure was fully robotized with a teleoperation architecture providing stable force feedback and visual feedback capabilities have been proposed for multi arms systems. This architecture was specifically developed for data collection and was used for artificial intelligence training. Moreover, the results obtained have been extended for building a trilateral teleoperation architecture for training purposes. In the following, the work concentrated on developing

the planning and coordination system for the control of multiple robotic instruments. The proposed system aims to deal with the uncertainty provided by the cognitive module and to generate a safe collision-free autonomous motion. Thanks to the experience acquired during the development of the previous strategies, a robotic assistance architecture for the renal access procedure during Percutaneous nephrolithotomy has been developed. The last problem addressed in the thesis is the control of the remote center of motion for robotic systems without mechanical remote center of motion. The proposed architecture considers the effects this constraint causes on the dynamics of the system and proposes a controller that aims to deal with the constraint maximizing flexibility and performance. This controller is also proposed within a teleoperation setup. All the proposed systems have been experimentally validated on physical setups, with the aim of verifying and confirming the applicability and the effectiveness of each of them.

Sommario

La robotica chirurgica può essere considerata ormai una realtà consolidata in campo medico. L'esempio più eclatante è sicuramente il robot chirurgico da Vinci®, rilasciato da Intuitive Surgical quasi 20 anni fa. Si stima che, solamente nel 2012, con il da Vinci®siano stati effettuati più di 200.000 interventi chirurgici, la maggior parte dei quali per isterectomie e asportazioni di prostata. Al giorno d'oggi, la maggior parte delle operazioni chirurgiche robotizzate vengono eseguite utilizzando tecniche di teleoperazione, nelle quali i movimenti del robot vengono direttamente controllati dal chirurgo, il quale siede ed opera da una console remota. In questo tipo di operazione il robot non possiede alcuna capacità di prendere decisioni o eseguire compiti in modo autonomo. Quest'ultimo è proprio uno dei principali temi sul quale si concentra oggi la ricerca in robotica chirurgica. L'autonomia, infatti, può offrire importanti vantaggi, che includono una maggiore efficienza e ripetibilità grazie al controllo sub-millimetrico del robot, una migliore qualità di esecuzione grazie al monitoraggio di segnali biometrici in tempo reale e alla guida assistita da computer, e minori costi. Un altro fattore che ha spinto in questa direzione sono stati sicuramente gli importanti sviluppi in tema di intelligenza artificiale, che hanno caratterizzato gli ultimi decenni. Tuttavia, è necessario tenere in considerazione che rendere autonomo un sistema robotico-chirurgico apre nuove importanti sfide in termini di controllo. Il sistema deve essere infatti in grado di, ad esempio, percepire l'ambiente circostante per cooperare con altri sistemi evitando collisioni, comprendere le dinamiche dell'ambiente per interagire con esso e compensare i disturbi esterni, e rispettare tutti i vincoli necessari ad eseguire correttamente la procedura chirurgica. Un altro problema importante quando si rende autonomo un sistema di questo tipo è la raccolta dei dati necessari per l'addestramento del sistema stesso, che spesso sono pochi e difficili da acquisire. Il lavoro di questa tesi inizia ad affrontare alcuni di questi difficili problemi ed emersi nello sviluppo del progetto Europeo Smart Autonomous Robotic Assistant Surgeon. Il primo passo è stato quello di convertire una procedura laparoscopica standard in una completamente robotizzata, realizzando un'architettura di teleoperazione per applicazioni a più bracci robotici, in grado di fornire feedback visivo e feedback di forza

stabile. Questa architettura è stata sviluppata specificamente per la raccolta dei dati ed è stata utilizzata per l'addestramento della parte relativa all'intelligenza artificiale. Inoltre, i risultati ottenuti sono stati estesi realizzando un'architettura di teleoperazione trilaterale volta al training. Successivamente, il lavoro si è concentrato sullo sviluppo del sistema di pianificazione e coordinamento per il controllo di più strumenti robotizzati. Il sistema proposto mira a gestire l'incertezza fornita dal modulo cognitivo e a generare un movimento autonomo sicuro e privo di collisioni. Grazie all'esperienza acquisita durante lo sviluppo delle strategie precedenti, è stata poi sviluppata un'architettura di assistenza robotica per la procedura di accesso renale durante la nefrolitotomia percutanea. L'ultimo problema affrontato in questo lavoro di tesi è stato il controllo del centro di istantanea rotazione per sistemi robotici senza centro di istantanea rotazione meccanico. L'architettura proposta considera gli effetti che questo vincolo provoca sulla dinamica del sistema e propone un controllore che mira a gestire il vincolo massimizzando flessibilità e prestazioni. Questo controllore è stato poi proposto anche all'interno di un'architettura di teleoperazione. Tutti i sistemi proposti sono stati validati sperimentalmente su set-up fisici, con l'obiettivo di verificare e confermare l'applicabilità e l'efficacia di ciascuno di essi.

Contents

List of Figures

List of Tables

List of Abbreviations

AI **A**rtificial **I**ntelligence.

AR **A**ugmented **R**eality.

COG **C**enter **O**f **G**ravity.

CT **C**omputed **T**omography.

DMDS **D**ual-**M**aster **D**ual-**S**lave.

DMSS **D**ual-**M**aster **S**ingle-**S**lave.

DOFs **D**egrees **O**f **F**reedom.

FDA **F**ood and **D**rug **A**dministration.

HMD **H**uman **M**ounted **D**isplay.

MMMS **M**ulti-**M**aster **M**ulti-**S**lave.

MPC **M**odel **P**redictive **C**ontroller.

MRI **M**agnetic **R**esonance **I**mages.

OR **O**perating **R**oom.

PC **P**assivity **C**ontroller.

PCNL **P**er**C**utaneous **N**ephro**L**ithotomy.

PO **P**assivity **O**bserver.

PTDPC **P**ower-based **T**ime **D**omain **P**assivity **C**ontrol.

RAMIS **R**obotic-**A**ssisted **M**inimally **I**nvasive **S**urgery.

RCM **R**emote **C**entre of **M**otion.

SAL **S**pectral **A**rc **L**ength.

SARAS **S**mart **A**utonomous **R**obotic **A**ssistant **S**urgeon.

SMMS **S**ingle-**M**aster **M**ulti-**S**lave.

SMSS **S**ingle-**M**aster **S**ingle-**S**lave.

TDPC **T**ime **D**omain **P**assivity **C**ontrol.

VR **V**irtual **R**eality.

List of Publications

Journal papers

- M. Minelli, N. Piccinelli, F. Falezza, F. Ferraguti, R. Muradore, and C. Secchi, "Two-layer based multi-arms bilateral teleoperation architecture," Submitted to IEEE Transactions on Control System Technology
- A. Leporini, E. Oleari, C. Landolfo, A. Sanna, A. Larcher, G. Gandaglia, N. Fossati, F. Muttin, U. Capitanio, F. Montorsi, *et al.*, "Technical and functional validation of a teleoperated multirobots platform for minimally invasive surgery," *IEEE Transactions on Medical Robotics and Bionics*, vol. 2, no. 2, pp. 148–156, 2020
- M. Minelli, F. Loschi, F. Ferraguti, and C. Secchi, "A two-layer trilateral teleoperation architecture for mentoring in surgical training," *IFAC-PapersOnLine*, vol. 54, no. 19, pp. 267–274, 2021
- G. De Rossi, M. Minelli, S. Roin, F. Falezza, A. Sozzi, F. Ferraguti, F. Setti, M. Bonfè, C. Secchi, and R. Muradore, "A first evaluation of a multi-modal learning system to control surgical assistant robots via action segmentation," *IEEE Transactions on Medical Robotics and Bionics*, vol. 3, no. 3, pp. 714–724, 2021
- F. Ferraguti, M. Minelli, S. Farsoni, S. Bazzani, M. Bonfè, A. Vandanjon, S. Puliatti, G. Bianchi, and C. Secchi, "Augmented reality and robotic-assistance for percutaneous nephrolithotomy," *IEEE robotics and automation letters*, vol. 5, no. 3, pp. 4556–4563, 2020
- S. Puliatti, M. Amato, F. Ferraguti, M. Minelli, S. Farsoni, A. Eissa, M. Rizzo, L. Bevilacqua, M. Sighinolfi, C. Secchi, *et al.*, "A combined augmented reality and robotic system for assistance in percutaneous nephrolithotomy procedures," *European Urology Open Science*, vol. 32, p. S44, 2021
- ——, "A combined augmented reality and robotic system for assistance in percutaneous nephrolithotomy procedures," *European Urology Open Science*, vol. 20, p. S70, 2020
- M. Minelli and C. Secchi, "A torque controlled approach for virtual remote centre of motion implementation," Submitted to IEEE robotics and automation letters

- M. Amato, A. Eissa, S. Puliatti, C. Secchi, F. Ferraguti, M. Minelli, A. Meneghini, I. Landi, G. Guarino, M. C. Sighinolfi, *et al.*, "Feasibility of a telementoring approach as a practical training for transurethral enucleation of the benign prostatic hyperplasia using bipolar energy: a pilot study," *World journal of urology*, vol. 39, no. 9, pp. 3465–3471, 2021

Conference papers

- M. Minelli, F. Ferraguti, N. Piccinelli, R. Muradore, and C. Secchi, "An energy-shared two-layer approach for multi-master-multi-slave bilateral teleoperation systems," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 423–429

- M. Minelli, A. Sozzi, G. De Rossi, F. Ferraguti, F. Setti, R. Muradore, M. Bonfè, and C. Secchi, "Integrating model predictive control and dynamic waypoints generation for motion planning in surgical scenario," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 3157–3163

- M. Minelli, A. Sozzi, G. De Rossi, F. Ferraguti, S. Farsoni, F. Setti, R. Muradore, M. Bonfè, and C. Secchi, "Linear mpc-based motion planning for autonomous surgery," Submitted to 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2020

- G. De Rossi, M. Minelli, A. Sozzi, N. Piccinelli, F. Ferraguti, F. Setti, M. Bonfé, C. Secchi, and R. Muradore, "Cognitive robotic architecture for semi-autonomous execution of manipulation tasks in a surgical environment," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 7827–7833

- M. Minelli and C. Secchi, "Dynamic-based rcm torque controller for robotic-assisted minimally invasive surgery," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 733–740

Failure-Resilient Multi-Robot Network Topologies

Nowadays, multi robots systems are becoming more and more widespread. Examples include the exploration of hazardous environments, military reconnaissance or surveillance of restricted access areas. The absence of pre-installed infrastructures, the high autonomy and ability to fulfill tasks that are difficult to tackle by a single robot give to multi-agent systems high flexibility and wide range of applications. However, the nodes of the network are subject to failure, introducing uncertainty and unpredictability into the system that could lead to non-fulfillment of the task. For this reason, algorithms aimed at robustness and maintaining network connectivity have been developed. Because of problems related to the use of real communication networks, to the difficulty of implementing distributed localization systems and to the cost associated with multi-agent applications, the development of this algorithms take place, most of the time, from theoretical and simulation point of view only. Part of this PhD was dedicated to the development and experimental validation on real multi-agent systems of robustness and connectivity maintenance algorithms. This work is not reported in this document as it covered only a small part of the PhD, but produced the following publications:

- J. Panerati, M. Minelli, C. Ghedini, L. Meyer, M. Kaufmann, L. Sabattini, and G. Beltrame, "Robust connectivity maintenance for fallible robots," *Autonomous Robots*, vol. 43, no. 3, pp. 769–787, 2019
- L. Siligardi, J. Panerati, M. Kaufmann, M. Minelli, C. Ghedini, G. Beltrame, and L. Sabattini, "Robust area coverage with connectivity maintenance," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2202–2208
- M. Minelli, M. Kaufmann, J. Panerati, C. Ghedini, G. Beltrame, and L. Sabattini, "Stop, think, and roll: Online gain optimization for resilient multi-robot topologies," in *Distributed Autonomous Robotic Systems*. Springer, 2019, pp. 357–370
- M. Minelli, J. Panerati, M. Kaufmann, C. Ghedini, G. Beltrame, and L. Sabattini, "Self-optimization of resilient topologies for fallible multi-robots," *Robotics and Autonomous Systems*, vol. 124, p. 103384, 2020

Chapter 1

Introduction

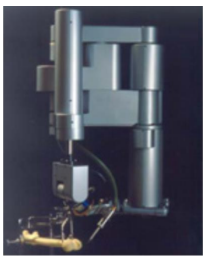
This chapter introduces the context on which this thesis develops and the reasons that guided the development of the proposed strategies. First, the surgical robotics field and the main challenges arising today are briefly introduced, focusing on those related to control when moving from teleoperated to autonomous systems. A general overview of the European-funded project Smart Autonomous Robotic Assistant Surgeon is then provided, project thanks to which it was possible to conduct much of the research activity and within which most of the work was developed and tested. Finally, an overview of the main contributions of this thesis and the structure of the document are reported.

1.1 Surgical Robotics

The introduction of robotics into the surgical world began in 1985 when a PUMA 560 (see Figure 1.1a) was used to increase the precision of positioning in CT-guided neurosurgical procedures [19]. Since then, several prototypes of surgical robots have been developed, with the aim of increasing the benefits for both the patient and surgeon. Some remarkable examples are:

- ROBODOC (Integrated Surgical Supplies Ltd., Sacramento, CA, USA) developed in 1992 and used in hip replacement surgeries [20]. First robot approved by the United States Food and Drug Administration [21] (FDA). See Figure 1.1b.
- AESOP 1000 (Computer Motion, Santa Barbara, CA, USA) developed in 1994 and used as a voice-controlled camera holder [22]. See Figure 1.1c.
- ZEUS (Computer Motion Inc., Goleta, CA, USA) developed in 1998 and used for coronary bypass anastomosis [23]. First robot introducing the idea of telesurgery. See Figure 1.1d.

(a) PUMA560



(b) ROBODOC

(c) AESOP1000

(d) ZEUS

Figure 1.1: Examples of surgical robots developed in the last decades.

In the last decades, robotic surgery has grown by leaps and bounds. The main reason is the benefits that surgical robotics can provide with respect to traditional surgical methods. For example, robots offer accuracy, stability, dexterity, easy integration with imaging technology, greater range of motion, and telesurgery capabilities, which are simply absent in traditional surgical methods.

The turning point came in 1997, when Integrated Surgical Systems (now Intuitive Surgical Inc., Sunnyvale, CA) [24] created the da Vinci®Surgical System, subsequently approved by FDA in July 2000 for general laparoscopic surgery. As reported in Figure 1.2, the da Vinci®Surgical System is composed of four robotic arms holding surgical tools (see Figure 1.2a) which are controlled by the surgeon through a remote console (see Figure 1.2b).

(a) da Vinci®Surgical System Robot **(b)** da Vinci®Surgical System Console

Figure 1.2: The da Vinci®Surgical System.

The da Vinci®robot is currently being used in various fields (urology, general surgery, gynecology, cardiothoracic, pediatric, etc) providing several advantages to conventional surgery such as motion scaling, immersive stereo 3D vision, and tremor compensation [25].

The advent of robotics in the medical field, and especially to the da Vinci®, led to a significant increase on the use of minimally invasive surgery, and, in particular, on the robotic-assisted minimally invasive surgery (RAMIS)[26], which is nowadays one of the most promising technique of robotic surgical procedure (see e.g. [27, 28]).

In RAMIS procedure, a small incision is made on the patient's abdominal wall through which the surgical tools are inserted and used to operate. Combining the advantages of manual minimally invasive surgery and those introduced by the use of robots, RAMIS has different advantages compared to open surgery. The patient benefits from reduced bleeding, pain, and recovery time thanks to the small incisions needed to perform the surgical operation. At the same time, the surgeon benefits from the robotic assistance and from an ergonomic posture to reduce fatigue and increase precision and dexterity.

For these reasons, there was a concrete interest from both the medical field and robot manufacturers in the development of these technologies, such that RAMIS is actually a consolidated and widespread reality.

Nowadays, the interest of the surgical community is focusing more and more on introducing autonomy. In fact, most robotic-assisted surgeries are carried out using teleoperation systems, such as the one provided by the da Vinci®: the surgeon sits at the remote console and directly controls the movements of the robotic arms. With this kind of technology, the robot is unable to perform tasks autonomously or make decisions, leaving the task execution and the entire cognitive load to the surgeon.

On this scenario, the introduction of autonomy can bring significant advantages: increased repeatability since the robot can perform precisely the same task in the same manner several times without fatigue, increased efficiency and reduced risk of human error thanks to the reduced cognitive load for the surgeon, easy biosignal feedback integration and computer-aided guidance to compute and/or correct robot behavior, and fewer costs due to reduced intervention time.

On the other hand, the development of autonomous surgical robotic systems opens up challenges from countless points of view. First of all, the robot must be able to perceive the surrounding environment. In the specific case of surgical procedures, it must be able to recognize the working region, avoid forbidden regions and distinguish the organs with which to interact from those with which avoid collisions. The robot must know precisely its kinematics and its constraints, to plan correctly compliant trajectory while guaranteeing safety for the patient. For example, in the RAMIS case, the movements of the robot must always be such that the point through which the tool is inserted is kept fixed. The robot must also be able to make decisions, depending on the operation is performing, from the phase the operation is and possibly react to unexpected events. Finally, the robot must be controlled such that decisions and planned trajectories are implemented correctly, making the robot able to effectively interact with the environment.

The results of this thesis are obtained addressing some of the challenging problems presented in the latter, and arising during the European funded project Smart Autonomous Robotic Assistant Surgeon (SARAS), where the goal is to develop a robotic system capable of autonomously performing surgical tasks in the context of RAMIS procedures.

1.2 SARAS

Nowadays, in surgical operations, several units of medical personnel are requested to stay in the operating room. For example, a typical laparoscopic intervention requires the presence of a main surgeon, of an assistant surgeon, of two nurses and of an anesthetist.

The introduction of robotic surgery with the da Vinci®robot has not decreased this number. In fact, during a robotized laparoscopic surgery, several units of medical personnel are also requested to stay in the operating room for supporting the main surgeon teleoperating the surgical robot. In particular, an assistant surgeon is always present for taking care of all the simple surgical procedures the leading surgeon cannot perform with the laparoscopic tools he/she is teleoperating (e.g. aspiration of blood in the operating area, removal of dead tissue after a cut and moving organs or tissue to make root for cutting or suturing) [29]. Figure 1.3 provides a clear example of this scenario.

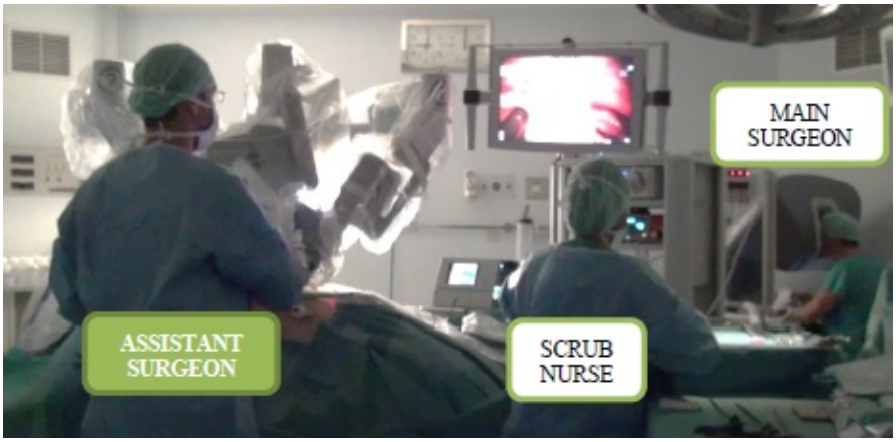


Figure 1.3: Current scenario in an operating room during a RAMIS procedure.

Although the assistant surgeon performs critical tasks only for 30% of the time of the surgical procedure and he/she has to stand, waiting, most of the time, it is common that an expert surgeon has to play the role of an assistant during an operation led by another surgeon. Considering the hourly cost of a surgeon, the current practice is very inefficient from an economic point of view. It is estimated that the 20% of the cost of the surgical intervention can be attributed to the assistant surgeon. Furthermore, the current practice is very inefficient also from a social point of view. In fact, both assistant and main surgeons need to rest for a fixed number of hours between interventions, reducing by a half the number of available surgeons (in particular for robotic interventions), leading to unnecessary long waiting lists.

In this context, the SARAS project aims at increasing the economic and social efficiency of the hospitals by developing a robotic surgical platform that allows the execution of RAMIS operations by only one surgeon. This will increase the efficiency and flexibility of operating room (OR) personnel, in particular surgeons, without affecting the quality of the surgical interventions.

The SARAS architecture needs to go beyond any existing systems for RAMIS. For these reasons, the development of the SARAS architecture has been split into three different platforms with three levels of increasing complexity. As the SARAS architecture need to be able to autonomously understand the surgical situation and to perform actions at the right place and time, the three platforms are used also to build the SARAS cognitive layer, with which the system is able to make decisions.

More details about the three SARAS platforms and the SARAS cognitive layer are reported in the next sections.

By incorporating prior knowledge of the workflow of the surgical procedure, SARAS aims also at reducing OR complications by monitoring the OR personnel during the procedure. The cognitive capabilities will allow also to address the very critical issue of surgical safety, a critical topic in the medical literature [30] and a main concern of the World Health Organization [31], and will contribute to the goal of reaching the "zero-accident vision in surgery" in the near future.

The SARAS architecture has been validated with two specific and very important robotic-assisted procedures, which cover almost 80% of all robotic interventions:

- radical prostatectomy, which is the resection of the whole prostate gland in male patients with prostate cancer, while preserving urinary continence and erectile function.
- partial or complete nephrectomy, which is the removal of the renal cell carcinoma, while keeping the healthy part of the kidney.

1.2.1 Platforms

As already mentioned, the development of the SARAS architecture has been split into three different platforms, with increasing level of complexity:

- the MULTIROBOTS-SURGERY platform.
- the SOLO-SURGERY platform.
- the LAPARO2.0-SURGERY platform.

MULTIROBOTS-SURGERY Platform

The SARAS Cognitive Layer is the brain of the entire SARAS system and is based on an artificial intelligence (AI) module, which requires a deep and extensive training phase. The purpose of this first platform is to acquire all the data necessary for this training phase.

In order to physically replace the assistant surgeon, the SARAS arms, a pair of robotic arms capable of handling the surgical instruments necessary for carrying out the operations, are developed. More details are reported in Section 1.2.3.

Then, a fully teleoperate architecture is developed, where the main surgeon seat at the da Vinci®console teleoperating the da Vinci®tools, and where the assistant surgeon teleoperates the two SARAS arms. In this scenario, the assistant surgeon performs the same actions as in standard robotic surgery, but teleoperating the tools instead of moving them manually. This architecture allows to acquire the movements of both surgeons, information used for the realization of the SARAS Cognitive Layer. An illustration of the SARAS MULTIROBOTS platform is reported in Figure 1.4.

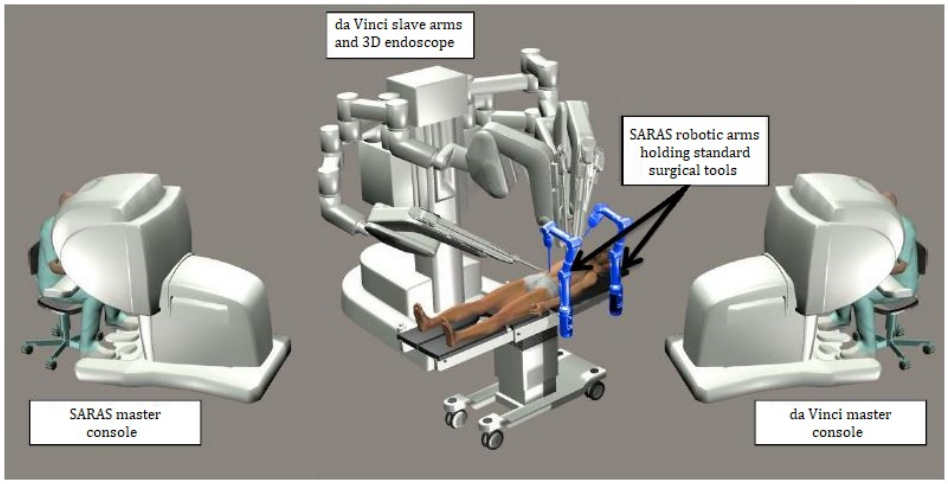


Figure 1.4: The SARAS MULTIROBOTS platform.

The platform is a bilateral Multi-Master Multi-Slave (MMMS) teleoperation system, where the two surgeons cooperate in a shared environment. The assistant surgeon console is equipped with a 3D Human Mounted Display (HMD), which provide the necessary visual feedback and virtual fixtures [32] (e.g. virtual walls avoiding the tools to reach forbidden regions or forces guiding the surgeon toward optimal paths during delicate phases of the procedure).

To get more realistic information, The MULTIROBOTS-SURGERY platform use accurate physical models of the human abdomen and pelvic region, i.e. phantoms manufactured using 3D printing and based on human Computed Tomography (CT), Magnetic Resonance Images (MRI) and ultrasound imaging.

SOLO-SURGERY Platform

The SARAS SOLO-SURGERY platform is the first autonomous SARAS platform, and play the role of the assistant to help the main surgeon at the da Vinci®console performing the surgical procedure. In this platform, the assistant surgeon is completely substituted by the SARAS system, allowing the main surgeon to perform the surgical procedure on its own.

On this platform, the available medical knowledge, the information acquired by the analysis of the data of the MULTIROBOTS-SURGERY platform and the nominal workflow of the surgical procedures are fused to make the first version of the SARAS Cognitive Layer. It is able to make decisions, act autonomously to help the main surgeon when deemed necessary, and receive, understand and execute explicit requests from the main surgeon.

With this platform, the main surgeon can interact with the SARAS system by vocal commands and by force/tactile feedback, with which can receive information on the interaction force.

Also in this case, the surgical procedures have been validated on advanced phantoms of the human abdomen and pelvic region.

An illustration of the SARAS SOLO-SURGERY platform is reported in Figure 1.5.

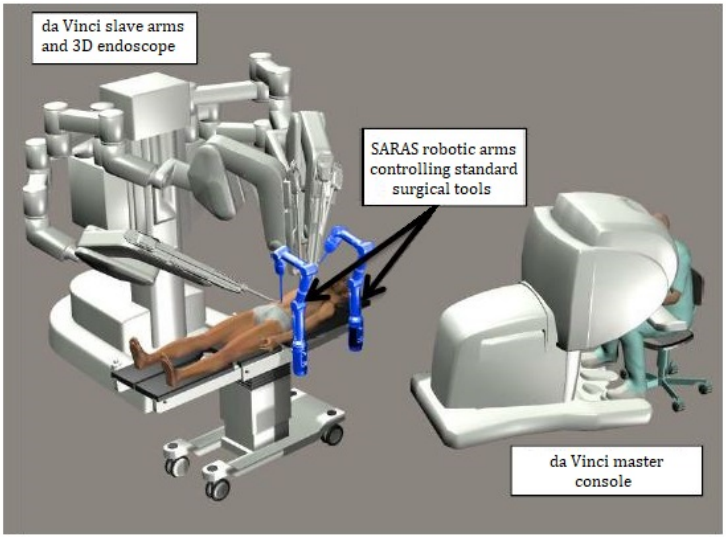


Figure 1.5: The SARAS SOLO-SURGERY platform.

LAPARO2.0-SURGERY Platform

In the last platform, the SARAS system plays the role of the assistant as in the SOLO-SURGERY case, but with the main surgeon using standard handheld laparoscopic tools. The idea of this platform is to make available the SARAS system also to hospitals that cannot afford to buy a da Vinci®-like surgical robotic system.

Since the main surgeon cannot rely on the da Vinci®console and its Augmented Reality (AR) and force-feedback features, the vocal commands (from the surgeon to the SARAS system) and the vocal/acoustic feedback (from the SARAS system to the surgeon) are enhanced. A visual interface is also developed to provide the surgeon with the necessary information.

Figure 1.6 illustrates the SARAS LAPARO2.0-SURGERY platform.

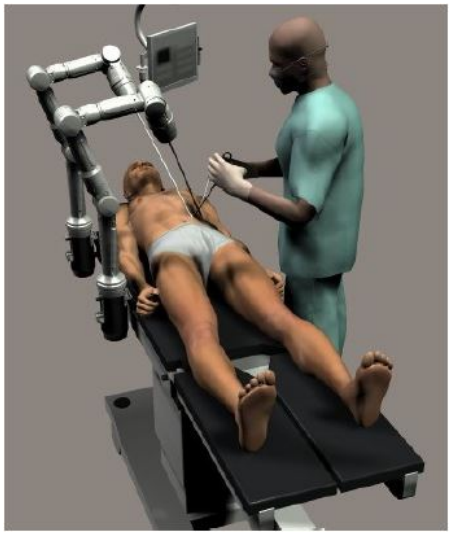


Figure 1.6: The SARAS LAPARO2.0-SURGERY platform

1.2.2 Cognitive Layer

As mentioned in the previous section, the SARAS architecture leverage on a Cognitive Layer base on an AI module, which require the development of new technologies in terms of 1. Human-Robot Interface and Interaction, 2. Perception, 3. Cognitive Control, and 4. Planning and Navigation. These technologies form the SARAS's cognitive layer for decision making reported in Figure 1.7 and described in the following.

Perception

The Perception module takes care of understanding the surgical area. Intra-operative images obtained by the 3D vision system of the da Vinci®endoscope (MULTIROBOTS-SURGERY and SOLO-SURGERY platforms) or by a surgical endoscope (LAPARO2.0-SURGERY platform) are fused with pre-operative information (3D models obtained from the elaboration of either CT or MRI of the patient before insufflation) to reconstruct a

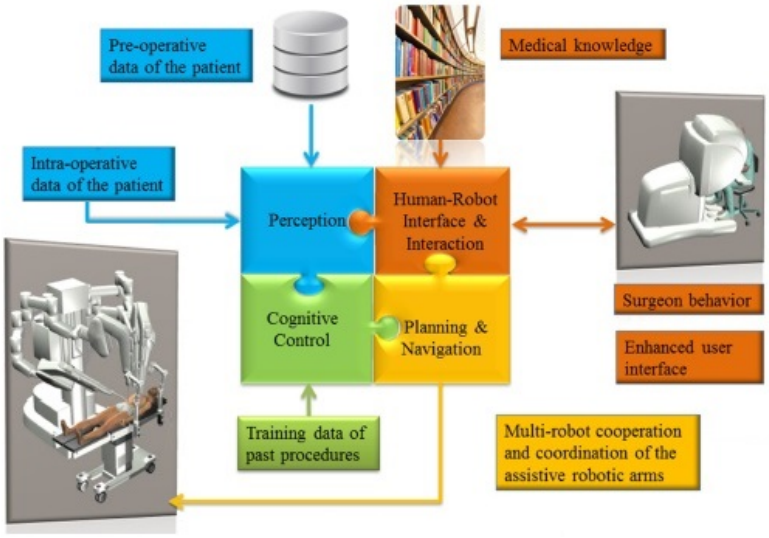


Figure 1.7: SARAS cognitive module architecture.

dynamic 3D model of the surgical cavity. This is used by the cognitive module to make inferences on the status of the surgical procedure and predictions on its future evolution.

Human-Robot Interface and Interaction

A natural and intuitive human-robot interface is provided by the SARAS system, allowing the surgeons to perform RAMIS by themselves with less training and lower mental fatigue. The interface provides also advanced and integrated force/tactile feedback (a functionality currently not provided by the da Vinci®), an integrated console with 3D vision, virtual fixtures, no-go regions, compensation of physiological movements, suggestions about the surgical procedure, and updated workflow and interventional checklist (not available nowadays).

Cognitive Control

Based on the outcomes of the Perception module, SARAS needs to perform sophisticated cognitive functions such as 1. recognize in real-time what the surgeon is doing, in order to react appropriately, 2. be aware of what the current stage of the procedure is, 3. based on its current estimate of the procedure's stage and of the action currently performed by the main surgeon, generate sensible guesses on what the latter is going to do next, to assist them in the most efficient way, 4. whether anomalous events (e.g. erratic behavior or proximity of sharp instruments to sensitive organs) are taking place, and, 5. based on all these elements, make a decision on what course of action to implement next (e.g. moving the robotic arms to assist the surgeon, adjusting camera view, issue a warning etc.)

As anticipated in the previous section, the cognitive module is trained from real intervention data. This allows the system to learn the structure of complex minimally invasive procedures, to identify anomalies, to understand the surgeon's actions (situation awareness), and his/her future needs (decision making).

Planning and Motion

In the SOLO-SURGERY and LAPARO2.0-SURGERY platforms, SARAS's assistive robotic arms move within the patient body, interact with anatomical structures and cooperate with the main surgeon. Once the SARAS system has interpreted the scene, made decisions and selected the actions to be performed to assist the surgeon, the Planning and Motion module translates these decisions into appropriate trajectories for the surgical tools mounted on SARAS arms. This allows the system to cooperate with the surgeon's surgical tools, teleoperated using da Vinci®'s slave arms (SOLOSURGERY) or manually moved (LAPARO2.0-SURGERY).

1.2.3 Robotic Arms

To substitute the assistant surgeon, the SARAS architecture includes two assistive robotic arms able to implement the tasks currently done by the assistant surgeon. They are composed of two mechatronic sub-systems:

- the Surgical Base System.
- the Surgical Active System.

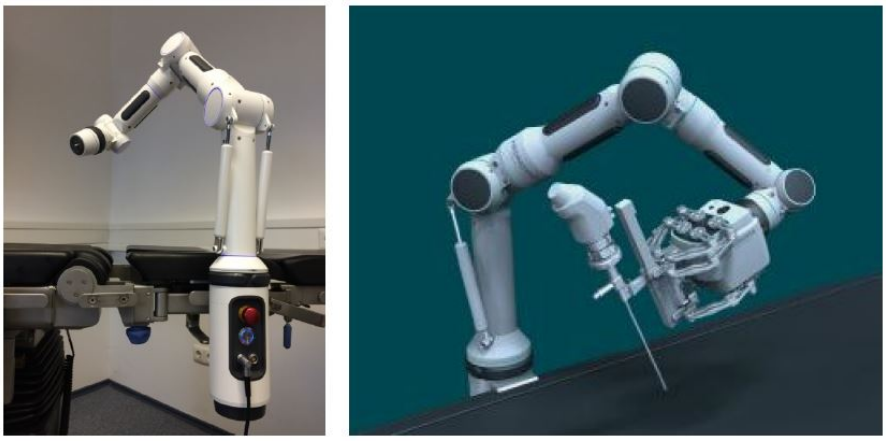


Figure 1.8: Concept of one of the SARAS arms.

The Surgical Base System is a seven Degrees of Freedoms (DOFs) passive holding arm attached to the surgical table. Using an integrated clamp system, the arm can be attached directly to the rail system of the OR table and can be aligned manually by the OR personnel according to the specific surgical procedure. With a payload of 2.5 kilograms, the goal of this first sub-system is holding in a specific position the second and most important device, the Surgical Active System. This second device is the one that holds the surgical instruments and takes care of their handling.

The development of the two assistive SARAS robotic arms start from the commercial system developed by one of the project's partners and focus on the hardware development of the dedicated active end-effector for the Surgical Base System, in order to make possible the handling minimally invasive surgery instruments.

Figure 1.8 illustrate the Surgical Base System and an example of the Surgical Active System.

Particular attention is paid to the number of DOFs the active system provides to the surgical tools. In fact, the insertion of the tools through the incision point on the patient body causes the movement of the tools to be constrained: only rotations around the three axes of the penetration point and a translation along the tool axis are permitted to not damage the patient body. The resultant tools motion from these constraints is called the Remote Center of Motion (RCM), and the Surgical Active System of the SARAS assistive arms must take it into account.

1.2.4 Consortium

The SARAS team is composed of the following partners, with different tasks and responsibilities:

- **University of Verona**
  Development of the Cognitive Control module, with particular focus on the autonomous execution of the surgical actions.
- **University of Modena and Reggio Emilia**
  Development of the Human-Robot Interface and Interaction module, with particular focus on Multi-Modal Human-Robot Interface and the MMMS Bilateral Teleoperation system for the SARAS MULTIROBOTS platform.
- **University of Ferrara**
  Development of the Planning and Navigation module, with particular focus on robot Dynamic Motion Planning.
- **Ospedale San Raffaele**
  Definition of the medical specifications, requirements, and risks analysis for the Human-Robot Interface architecture, providing medical competence of all kinds throughout the project. It is also responsible for the validation of all the Human-Robot Interface platforms.
- **Universitat Politècnica de Catalunya**
  Development of the Planning and Navigation module, with particular focus on Multi-Robot Cooperation and Task Identification and Planning.
- **University of Dundee**
  Analysis of anatomical parts and developments of artificial and hybrid phantoms used for the validation of the SARAS architecture.
- **Oxford Brookes University**
  Development of the Perception and Cognitive Control modules, with particular focus on surgeon verbal command recognition, tracking of deformable organs and detection, segmentation, and labeling of scene elements.
- **Medineering GmbH**
  Hardware and Software development of the SARAS arms Surgical Active System.
- **ACMIT GmbH**
  Development of augmented phantoms used for the validation of the SARAS architecture.

1.3 Contribution and Thesis Outline

This thesis focuses on the development of novel control strategies to address some of the problems arising when moving from standard surgical systems, most of which are teleoperation systems, to autonomous surgical systems.

The main contributions of this thesis are:

- a novel two-layer architecture for MMMS teleoperation systems based on the concept of shared energy tank, that allows operations while guaranteeing high fidelity and transparent behavior.
- a novel trilateral teleoperation control architecture for surgical training purposes, guaranteeing stability and high flexibility.
- a Model Predictive Controller (MPC) that computes collision-free trajectory of multiple surgical tools, and a novel strategy to compute suitable waypoints able of improving the convergence of the motions towards the target positions, while avoiding obstacles.
- an innovative solution, based on an AR application combined with a robotic system, that can assist both an expert surgeon in improving the performance of the surgical operation and a novel surgeon in strongly reducing his/her learning curve, performing the renal access during Percutaneous Nephrolithotomy (PCNL).
- a novel formulation of a dynamic-based torque controller for the implementation of virtual RCM for RAMIS that guarantees the RCM constraint while allowing the user to freely design the desired behavior.
- an experimental validation of all the proposed architectures.

The rest of the thesis is organized as follows.

Chapter 2 reports the development of the bilateral MMMS teleoperation system for the SARAS MULTIROBOTS platform. Problems related to passivity and transparency are addressed when communication takes place by means of a delayed communication channel. Therefore, the two-layer approach is extended to the case of MMMS systems by proposing the novel concept of shared energy tank. The results obtained after a first development of the architecture are extended in order to improve the use of energy, reducing the conservativeness of the system and allowing a more transparent behavior. The overall architecture is then validated experimentally on the SARAS setup. Finally, the presented strategy is extended and validated on a trilateral DMSS teleoperation setup.

Chapter 3 reports the development of the Planning and Navigation module of the SARAS cognitive layer. The motion planning of multiple surgical tools operating in the same workspace is considered when constraints on the maximum velocity of the surgical tools and on the collisions between the tools need to be guaranteed. The problem related to the accuracy of the action recognition of the SARAS cognitive layer is also addressed including a confidence level in the constraints formulation. The results obtained after a first validation of the controller are extended in order to improve the performance of the system and to allow a real-time implementation. The controller is finally validated on a simulated setup and then experimentally on the SARAS setup.

Chapter 4 reports the development of a robotic assistance architecture for the renal access procedure during PCNL. In the pre-operative phase, the model of the anatomical parts involved in the procedure are segmented and converted to a 3D model, with which the insertion trajectory is planned. In the intra-operative phase, the surgeon visualizes the 3D model and the planned trajectory by means of an AR headset and performs the renal access with the assistance of the robot, on which the surgical tool is mounted. The system is validated through experiments performed by a sample of multiple users.

Chapter 5 reports the development of a torque control strategy for the implementation of virtual RCM on light-weight torque controlled manipulators. The dynamic model of the manipulator subject to the RCM is formulated and exploited to synthesized the control action, leaving the user the possibility to freely design the manipulator behavior. The results obtained after a first validation of the controller are extended in order to improve the capabilities of the controller to implement compliance motion control and make the robot effectively interact with the environment. The controller is then validated on a simulated setup and then experimentally on a real torque-controlled manipulator. Finally, the controller is extended to admittance controlled manipulator and experimentally validated on a teleoperation setup.

Chapter 6 reports the conclusions of the entire work of thesis and the future direction on which researches could be driven.

Chapter 2

Passivity based Bilateral Teleoperation Architecture

This Chapter reports the development of the bilateral MMMS teleoperation system for the SARAS MULTIROBOTS platform. Problems related to passivity and transparency are addressed when communication takes place by means of a delayed communication channel. Therefore, the two-layer approach is extended to the case of MMMS systems by proposing the novel concept of shared energy tank. The results obtained after a first development of the architecture are extended in order to improve the use of energy, reducing the conservativeness of the system and allowing a more transparent behavior. The overall architecture is then validated experimentally on the SARAS setup. Finally, the presented strategy is extended and validated on a trilateral DMSS teleoperation setup.

The work presented in this chapter is published in [10, 1, 2, 3].

2.1 Multi-Master Multi-Slave Bilateral Teleoperation Architecture

2.1.1 Introduction

Bilateral teleoperation systems allow to extend the reach of human beings allowing to interact with a remote environment while locally feeling the interaction force.

The standard architecture of a teleoperation system includes a local device, the master, and a remote device, the slave. The master device is located at the operator side and is used to capture and send the operator movements to the slave device, in the form of pose information. The slave device is located at the remote environment and replicates the motion of the master device, performing the task.

In a bilateral teleoperation architecture, the interaction between the slave device and the environment is measured or estimated and sent back to the master device, in the form of force feedback. The master device is then able to replicate the interaction with the environment, providing the user with the feeling of being directly interacting with the remote environment. Bilateral teleoperation has been used for many applications like manipulation of chemical and nuclear material substances [33], for bomb disposal [34], or surgical procedures [35]. An example of a teleoperation system for military applications is reported in Figure 2.1.

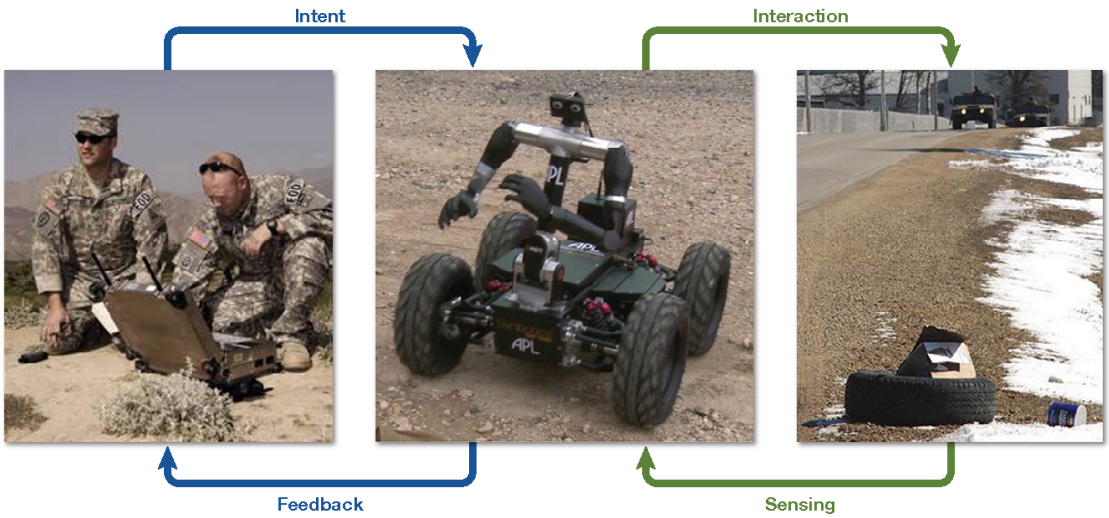


Figure 2.1: Example of teleoperation system for military applications

When dealing with complex tasks, a teleoperation architecture composed of a single master device and a single slave device (SMSS) may not provide the necessary dexterity and flexibility for accomplishing a task on the remote environment. In these cases, a MMMS teleoperation architecture can provide the desired level of remote mobility and interaction capabilities. A well-known example of application of this kind of teleoperation architecture is the da Vinci®Surgical System. The surgeon uses two hand-held haptic interfaces to teleoperate three or four arms of the surgical robot for performing complex tasks.

Stability and transparency are the main issues when controlling a bilateral teleoperation system. Due to the distance between the devices, the exchange of information between the local and the remote sides typically happens over a delayed communication channel. The communication delays and the interaction with a poorly known environment are the main sources of instability in any teleoperation architecture. Several control algorithms have been proposed in the literature to solve the stability problem. However, these algorithms typically decrease the transparency of the system, i.e. the measure of how well the desired motion and force feedback are implemented at the remote and local sides, respectively [36].

Starting from [37], the two-layer framework has been extended to MMMS teleoperation architecture. A straightforward generalization would be to decompose the MMMS system into pairs of master-slave robots and to implement the standard two-layer architecture on each pair. This solution is conceptually very simple but it would imply the implementation of a coordination strategy among the transparency layers. Furthermore, the presence of many energy tanks may lead to energetic inconsistencies that may cause an excess of conservatism and degrade the performance of the MMMS teleoperation system. Finally, in general, the number of master robots may be different from the number of slaves and, therefore, a pairwise decomposition may not be possible.

To avoid these problems and to keep the simplicity of the two-layer architecture, the use of only two shared energy tanks is proposed, one for the masters and the other for the slaves. One passivity layer will manage the passive exchange of energy between the tanks and between each tank and the robots. One transparency layer will compute the desired behavior for the whole multi-master side and for the multi-slave side. This control architecture will minimize the conservatism of the teleoperation system while allowing to consider the multi-arms system as a whole.

2.1.2 Related works

Several control architectures have been proposed for implementing a bilateral teleoperation system [38]. This section focus on the works that specifically address the problems arising in the considered architecture: the design of a stable and transparent MMMS teleoperation architecture with respect to time delays and to the interaction with poorly known environments.

Passivity-based control strategies have been proven to be very successful since they allow to robustly handle the interaction with unstructured environments and to compensate the destabilizing effects of the communication delay.

For example, the wave variables developed by Niemeyer *et al.* [39] are one of the main tools used to achieve a stable teleoperation system and have been exploited for decades. Based on a given scheme, the wave variables encrypt the power variables (velocities and forces) exchanged between the local and the remote sides to turn the communication channel into a passive element, regardless of the time delays. Moreover, if both sides are passive, the overall teleoperation architecture is passive too and thus stable.

An example of MMMS application of this concept has been developed by Huang *et al.* [40]. Based on the forward wave compensation method [41], a backward wave compensation method, and an energy regulator [42], they developed a dual-master dual-slave (DMDS) teleoperation system. An asymmetric compensation method enhances the velocity and force tracking performances while ensuring the passivity of the system.

The main drawback of wave variables is that the inherent dynamics of wave-based communication channels is often deleterious for the transparency of the teleoperation system.

Starting from the Time Domain Passivity Control (TDPC) algorithm developed by Hannaford *et al.* [43], Ryu *et al.* [44] proposed an application to bilateral telemanipulation. In their approach, two elements are introduced: the Passivity Observer (PO) and the Passivity Controller (PC). The PO monitors the energy flow into the system and a time-varying damping element, the PC, is activated to dissipate the excess energy when necessary. An improved version of this kind of architecture, the Power-based Time Domain Passivity Control (PTDPC), has been proposed by Ye *et al.* [45], where the power flow, rather than the energy flow, is monitored to achieve a smoother activation of the PC.

An example of MMMS implementation of the PTDPC has been developed by Chen *et al.* [46]. In particular, the SMSS PTDPC architecture is extended to solve the passivity problem in an MMMS scenario and a novel communication structure allows the system to deal with the complexity of the communication channel when multiple local and remote devices are interconnected. The main drawback of these kinds of approaches is related to the conservativeness of the resulting system, often too high and deleterious for transparency.

Alternative approaches are based on the idea of predicting the non-delayed output of the plant by exploiting a model of the system and compensating the problem introduced by the delays. Smith *et al.* first proposed a linear predictive controller [47] known as *Smith predictor*. In a teleoperation system, the master and the slave devices are haptic interfaces or robotic manipulators, which model is typically non-linear and may also vary with time (e.g. in case of user interaction or object picking). Huang *et al.* [48] introduced a recurrent neural network to capture the remote robot's non-linearity and integrated it with linear Smith predictor in order to improve the performance of the system.

With the use of a Slotine-Li adaptive control algorithm, Fite *et al.* [49] developed an architecture that can also deal with time-varying environment dynamics. Smith *et al.* [50] introduced the online training of the network, allowing the system to estimate and map the remote device and environment dynamics at the local side. This increases the usability of the system, especially in the presence of substantial delays in the communication channel. The online knowledge of the remote and environment dynamics allows the system to work also if the environment dynamics is nonlinear and time-varying.

These techniques are very promising but, unfortunately, examples of applications can be found only for trilateral scenarios, as proposed by Li *et al.*[51] and by Ji [52], with particular focus on the Dual-Master Single-Slave (DMSS) teleoperation architecture.

Franken *et al.* [37] developed a teleoperation control architecture based on a two-layer framework. By exploiting the concept of energy tank, this kind of approach splits the control architecture into two hierarchical layers. The higher layer is used to implement a strategy that addresses the desired transparency while the lower layer ensures that passivity is not violated.

An example of application of this kind of technique for a SMSS teleoperation architecture has been developed by Ferraguti *et al.*[35], where the two-layer framework has been exploited to passively implement the SMSS bilateral teleoperation architecture and to compensate the position mismatch between the local and the remote device. Extensions to Single-Master Multi-Slave (SMMS) teleoperation systems were also proposed, for example by Secchi *et al.* [53].

2.1.3 Problem Statement

Consider a system composed by N_m masters and N_s slave robots, fully actuated and locally gravity compensated. Each robot can be modeled as the following n-DOFs Euler-Lagrange system:

$$\Lambda_{w_i}\left(x_{w_i}(t)\right)\ddot{x}_{w_i}(t)+\mu_{w_i}\left(x_{w_i}(t),\dot{x}_{w_i}(t)\right)\dot{x}_{w_i}(t)=F^{\tau}_{w_i}(t)+F^{ext}_{w_i}(t) \tag{2.1}$$

where $w\in\{m,s\}$, where the subscripts m and s indicate the master and the slave side, respectively, and $i=1,\ldots,N_w$, $x_{w_i}(t)\in\mathbb{R}^n$ are the coordinates of the configuration of the end-effector in the task space, $\Lambda_{w_i}\left(x_{w_i}(t)\right)\in\mathbb{R}^{n\times n}$ is the symmetric and positive-definite inertia matrix and $\mu_{w_i}\left(x_{w_i}(t),\dot{x}_{w_i}(t)\right)\in\mathbb{R}^{n\times n}$ is the Coriolis/centrifugal matrix. The term $F^{\tau}_{w_i}(t)\in\mathbb{R}^n$ represents the control inputs while $F^{ext}_{wi}(t)\in\mathbb{R}^n$ is the vector of generalized external forces, i.e. the force applied by the user or the force applied by the environment.

For ease of notation all the robots are considered to have the same number of DOFs. All the results can be easily generalized to the case where the robots have a different number of DOFs.

It is possible to build an Euler-Lagrangian model of the whole master and slave sides. Defining:

$$\Lambda_w\left(x_w(t)\right)=\mathrm{diag}(\Lambda_{w_1},...,\Lambda_{w_{N_w}}) \tag{2.2}$$

$$\mu_w\left(x_w(t),\dot{x}_w(t)\right)=\mathrm{diag}(\mu_{w_1},...,\mu_{w_{N_w}}) \tag{2.3}$$

$$x_w(t)=\begin{bmatrix}x_{w_1}(t)\\ \vdots\\ x_{w_{N_w}}(t)\end{bmatrix} \tag{2.4}$$

$$F^{\tau}_w(t)=\begin{bmatrix}F^{\tau}_{w_1}(t)\vdots\\ F^{\tau}_{w_{N_w}}(t)\end{bmatrix} \tag{2.5}$$

$$F_w^{ext}(t) = \begin{bmatrix} F_{w_1}^{ext}(t) \\ \vdots \\ F_{w_{N_w}}^{ext}(t) \end{bmatrix} \tag{2.6}$$

and exploiting (2.1), each side of the teleoperation system can be modeled as the following Euler-Lagrange system:

$$\Lambda_w(x_w(t))\ddot{x}_w(t) + \mu_w(x_w(t), \dot{x}_w(t))\dot{x}_w(t) = F_w^{\tau}(t) + F_w^{ext}(t) \tag{2.7}$$

The kinetic energy of the system described in (2.7) is given by the sum of the kinetic energies of all the robots in the w side and, compactly, it is defined as:

$$V_w\left(\dot{x}_w(t)\right) = \frac{1}{2}\dot{x}_w(t)^T \Lambda_w\left(x_w(t)\right)\dot{x}_w(t) \tag{2.8}$$

With a slight abuse of notation, the terms $V_w(t)$, $\Lambda_w(t)$ and $\mu_w(t)$ will be hereafter used to indicate, respectively, the value of $V_w\left(\dot{x}_w(\,t)\right)$, $\Lambda_w\left(x_w(t)\right)$ and $\mu_w\left(\dot{x}_w(t), x_w(t)\right)$ at time t. Moreover, for clarity of presentation, the time-dependence of the variables will be omitted when the context is clear.

The passivity of the multi-robot system, either at the master or at the slave sides, can be easily shown. In fact, from (2.8) it follows that:

$$\dot{V}_w(t) = \dot{x}_w(t)^T \Lambda_w(t)\ddot{x}_w(t) + \frac{1}{2}\dot{x}_w(t)^T \dot{\Lambda}_w(t)\dot{x}_w(t) \tag{2.9}$$

Computing $\ddot{x}_w(t)$ from (2.7), replacing it in (2.9) and considering that $\dot{\Lambda}_w(t) - 2\mu_w(t)$ is skew-symmetric one obtains:

$$\dot{V}_w(t) = \dot{x}_w^T(t)\left(F_w^{\tau}(t) + F_w^{ext}(t)\right) \tag{2.10}$$

which implies that:

$$\int_0^t \dot{x}_w^T(\tau)\left(F_w^{\tau}(\tau) + F_w^{ext}(\tau)\right)d\tau \geq -V_w(0) \tag{2.11}$$

which means that the system is passive.

The goal of the system is to implement the teleoperation architecture in a stable way, guaranteeing to the user a transparent and safe interaction with a poorly known environment (i.e. the human body) and to be flexible, in order to change the kind of feedback that should be provided to the user.

To passively implement the bilateral teleoperation architecture, an energy tank need to be introduced at both master and slave sides. Firstly introduced in [54] and then used in a wide range of applications like multi-robot systems [55] and surgical robotics [56], the energy tank is used to store the energy dissipated by a system and then to re-use this energy to implement the control action, without violating passivity.

Even if, as reported in the previous section, each side of the teleoperation is passive, when both sides of the teleoperation system are interconnected by means of the communication channel, this condition is not sufficient to guarantee the passivity condition of the whole system.

The introduction of the tank on each side of the teleoperation will allow, as will be clear in the next sections, to keep the entire system passive even after the two sides are interconnected.

2.1.4 Shared Energy Tanks

In order to be able to harvest some energy for filling the energy tank when necessary, a controlled dissipation is first implemented on each robot. Then, a shared energy tank is connected to all the robots on the same side. This can be done by splitting the control input of each robot into the sum of two terms:

$$F_{w_i}^{\tau} = \phi_{w_i}^{t} + \phi_{w_i}^{d} \tag{2.12}$$

The first term is exploited to implement a control action on the robot while the second term is exploited for implementing a variable local damping by setting:

$$\phi_{w_i}^{d} = -D_{w_i}(t)\dot{x}_{w_i}(t) \tag{2.13}$$

where $D_{w_i}(t) \in \mathbb{R}^{n \times n}$ is a time-varying positive semi-definite matrix. Embedding the damping injection into (2.1) returns the following damped Euler-Lagrangian model for each robot:

$$\Lambda_{w_i}\left(x_{w_i}\right)\ddot{x}_{w_i} + \mu_{w_i}\left(x_{w_i}, \dot{x}_{w_i}\right)\dot{x}_{w_i} + D_{w_i}\dot{x}_{w_i} = \phi_{w_i}^{t} + F_{w_i}^{ext} \tag{2.14}$$

A shared energy tank is then placed at each side of the MMMS teleoperation system. The energy tank is an energy storing element that can be represented by:

$$\begin{cases} \dot{x}_{t_w} = \dfrac{\sigma_w}{x_{t_w}} \displaystyle\sum_{i=1}^{N_w} \dot{x}_{w_i}^{T} D_{w_i}(t)\dot{x}_{w_i} + u_{t_w} \\ y_{t_w} = \dfrac{\partial T_w}{\partial x_{t_w}} \end{cases} \tag{2.15}$$

where $x_{t_w} \in \mathbb{R}$ is the state of the tank, $(u_{t_w}, y_{t_w}) \in \mathbb{R} \times \mathbb{R}$ is the power port through which the tank can exchange energy with the rest of the world and:

$$T_w(x_{t_w}) = \frac{1}{2}x_{t_w}^2 \tag{2.16}$$

is the energy stored in the tank.

Using (2.16) with (2.15) it is easy to see that:

$$\dot{T}_w(x_{t_w}) = \sigma_w \sum_{i=1}^{N_w} \dot{x}_{w_i}^T D_{w_i}(t) \dot{x}_{w_i} + u_{t_w} y_{t_w} \tag{2.17}$$

namely, the tank stores the energy dissipated by all the robots using the local damping injection and that energy can be injected/extracted from the port (u_{t_w}, y_{t_w}).

Each robot is interconnected to the energy tank in order to exploit the energy stored in the tank for implementing the desired input. This can be done by the following power preserving interconnection between all the robots and the shared energy tank:

$$\begin{cases} \phi_{w_i}^t = \omega_{w_i} y_{t_w} \\ u_{t_w} = -\sum_{i=1}^{N_w} \omega_{w_i}^T \dot{x}_{w_i} \end{cases} \qquad i = 1, \ldots, N_w \tag{2.18}$$

Since:

$$\sum_{i=1}^{N_w} \dot{x}_{w_i}^T \phi_{w_i}^t = -u_{t_w} y_{t_w} \tag{2.19}$$

it means that each robot can extract/inject energy from/in the tank in order to implement the desired input by properly choosing the modulation factor $\omega_{w_i} \in \mathbb{R}^n$. Since the tank is shared, the energy in the tank by some robots can be re-used by other robots for implementing non-dissipative actions. Thus, the multi-robot system manages the energy available in the tank as a single entity.

By grouping (2.14) and by considering (2.15) and (2.18), it is possible to model each side of the MMMS teleoperation system as:

$$\begin{cases} \Lambda_w \ddot{x}_w + \mu_w \dot{x}_w + D_w \dot{x}_w = \omega_w x_{t_w} + F_w^{ext} \\ \dot{x}_{t_w} = \dfrac{\sigma_w}{x_{t_w}} \dot{x}_w^T D_w \dot{x}_w - \omega_w^T \dot{x}_w \end{cases} \tag{2.20}$$

where:

$$D_w(t) = \mathrm{diag}(D_{w_1}(t), \ldots, D_{w_{N_w}}(t)) \tag{2.21}$$

and:

$$\omega_w = \mathrm{diag}(\omega_{w_1}, ..., \omega_{w_{N_w}}) \tag{2.22}$$

with the arguments of the functions have been omitted for ease of notation.

The term σ_w is a design parameter that is used to bound the energy stored into the tank:

$$\sigma_w = \begin{cases} 1 & \text{if } T(x_{t_w}) < T_w^{max} \\ 0 & \text{otherwise} \end{cases} \tag{2.23}$$

where T_w^{max} represents the energy upper bound. Indeed, it is necessary to avoid excessive energy storing in the tank that may allow the implementation of practically unstable behaviors [57].

If $x_{t_w(t)} = 0$, i.e. when no more energy is left in the tank, the (2.20) becomes singular.

To avoid this problem the tank is initialized with some energy and the energy extraction below a certain threshold is prevented, as described in [57]. Thus, $x_{t_w}(0)$ is chosen such that $T_w(x_{t_w}(0)) \geq T_w^{min}$, where $T_w^{min} > 0$ is the minimum amount of energy that needs to be stored in the tank.

As proposed in [37], designing the local damping of each robot as variable allows regulating the energy store into the tank more effectively, according to the working condition. Following this principle, the local damping is designed as:

$$D_w = \begin{cases} D_w^{min}, & \text{if } T_w(x_{t_w}) > T_w^{b_{max}} \\ \xi\big(T_w(x_{t_w})\big), & \text{if } T_w^{b_{min}} \leq T_w(x_{t_w}) \leq T_w^{b_{max}} \\ D_w^{max}, & \text{if } T_w(x_{t_w}) < T_w^{b_{min}} \end{cases} \tag{2.24}$$

If the energy in the tank exceeds the energy upper threshold $T_w^{b_{max}}$ the local damping injection of the robot is set to a minimum level D_w^{min} since harvesting energy is not needed. If the energy in the tank is going below the energy lower threshold $T_w^{b_{min}}$, the local damping injection of the robot is set to a maximum level D_w^{min}, in order to fill energy into the tank as quickly as possible. Otherwise $D_w = \xi(T_w(x_{t_w}))$, where $\xi(T) : \mathbb{R} \to \mathbb{R}^{(N_w n)\times(N_w n)}$ is any smooth non-increasing function such that $D_w = D_w^{min}$ if $T_w(x_{t_w}) = T_w^{b_{max}}$ and $D_w = D_w^{max}$ if $T_w(x_{t_w}) = T_w^{b_{min}}$.

The choice of $\xi(T_w(x_{t_w}))$ guarantees a smooth transition between D_w^{min} and D_w^{max}, without discontinuities in the forces applied to the devices. The constants T_w^{min}, T_w^{max}, $T_w^{b_{max}}, T_w^{b_{min}}, D_w^{min}, D_w^{max}$ are application-dependent design parameters. As defined in (2.24) all the robots contribute in the same way to the energy harvesting. Nevertheless, robot-specific damping strategies may be designed.

Finally, the desired input for the robots can be achieved by setting the modulating term as:

$$\omega_{w_i} = \begin{cases} \dfrac{F_{w_i}^d}{x_{t_w}} & \text{if } T(x_{t_w}) \geq T_w^R \\ K_w(T(x_w))\dfrac{F_{w_i}^d}{x_{t_w}} & \text{otherwise} \end{cases} \tag{2.25}$$

where:

$$K_w(T(x_{t_w})) = \max\left(0, \frac{T(x_{t_w}) - T_w^{min}}{T_w^R - T_w^{min}}\right) \tag{2.26}$$

thus, if there is enough energy in the tank, the desired input $F_{w_i}^d$ is implemented otherwise only a scaled version of the desired input is implemented. In the worst case $K_w(T(x_{t_w})) = 0$ and, therefore, nothing will be implemented in order to preserve passivity. Nevertheless, since the local damping is set to its maximum when $T(x_{t_w}) < T_w^R$, its very unlikely that $K_w(T(x_{t_w})) = 0$ in practice.

Figure 2.2 shows the coupling of two generic master or slave devices with the energy tank.

The augmented model in (2.20) consists of system (2.7) with the damping element $D_w(x_{t_w})$ and the tank that is energetically coupled through the input ω_{w_i}. The ki-

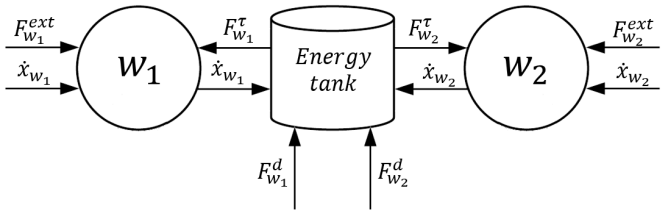


Figure 2.2: Coupling of two generic master or slave devices w_1 and w_2 with the energy tank.

netic energy is still given by (2.8). However, using the same procedure that leads from (2.8) to (2.10), for the augmented model it follows that:

$$\dot{V}_w(t) = \dot{x}_w^T \omega_w x_{t_w} + \dot{x}_w^T F_w^{ext} - \dot{x}_w^T D_w \dot{x}_w \tag{2.27}$$

Furthermore, by plugging (2.18) in (2.17) and by considering the definition of D_w in (2.20), it follows that:

$$\dot{T}_w(t) = \sigma_w(\dot{x}_w^T D_w \dot{x}_w) - x_{t_w} \omega_w^T \dot{x}_w \tag{2.28}$$

Each side of the teleoperation system augmented with the shared energy tank continues to remain a passive system, as proven in the following.

Proposition 1. *The system in* (2.20) *is passive with respect to the pair* $((F^{ext}_{w_1}, \ldots, F^{ext}_{w_{N_w}}), (\dot{x}_{w_1}, \ldots, \dot{x}_{w_{N_w}}))$

Proof. Consider as a storage function the total energy of the teleoperation system (2.20):

$$\mathcal{V}(t) = V_w(t) + T_w(t) \tag{2.29}$$

where $V_w(t)$ represents the energy associated to the master or slave device under consideration and T_w the energy stored in the tank. From (2.29) it follows that:

$$\dot{\mathcal{V}}(t) = \dot{V}_w(t) + \dot{T}_w(t) \tag{2.30}$$

Substituting (2.27) and (2.28) in (2.30) returns:

$$\dot{\mathcal{V}}(t) = \dot{x}_w^T F_w^{ext} - (1 - \sigma_w) \dot{x}_w^T D_w \dot{x}_w \tag{2.31}$$

Since $\sigma_w \in \{0, 1\}$, it follows that:

$$\dot{\mathcal{V}}(t) \leq \sum_{i=1}^{N_w} \dot{x}_{w_i}^T F_{w_i}^{ext} \tag{2.32}$$

which implies the following passivity condition:

$$\mathcal{V}(t) - \mathcal{V}(0) \leq \int_0^t \sum_{i=1}^{N_w} \dot{x}_{w_i}^T(\tau) F_{w_i}^{ext}(\tau) d\tau \tag{2.33}$$

□

2.1.5 The Bilateral Control Architecture

The overall architecture for the bilateral teleoperation of a multi-master-multi-slave system is shown in Figure 2.3 and it can be decomposed into two layers: a *Transparency Layer* and a *Passivity Layer*. In the figure, the example of two master devices ($N_m = 2$) and two slave devices ($N_s = 2$) is provided. In order to keep Figure 2.3 simple, the forces exchange is not shown in the picture.

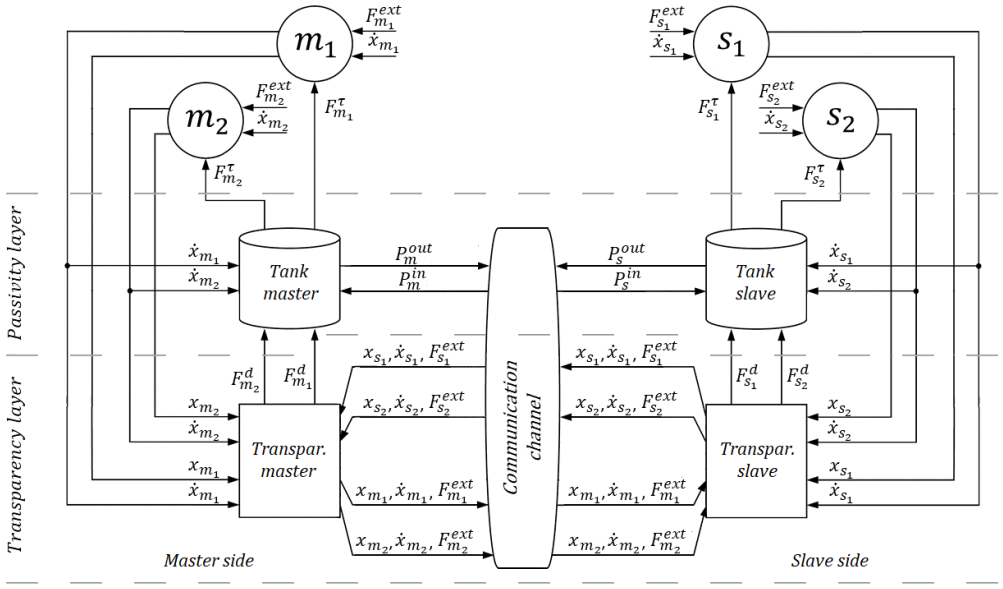


Figure 2.3: Coupling of two generic master devices m_1 and m_2 with two slave devices s_1 and s_2, and one tank per side by means of the communication channel

In the Transparency Layer, master and slave devices exchange position, velocity and force information that are used for computing the desired inputs (F^d_{m1}, F^d_{m2}, F^d_{s1}, F^d_{s2}). These forces are sent to the Passivity Layer, whose role is to passively implement them using the energy stored in the tanks.

In order to interconnect master and slave sides by means of a delayed communication channel, each tank is endowed with two power inputs, as proposed in [56]. Master and slave energy tanks can then exchange power for balancing the amount of energy stored at master and slave sides.

Formally, the overall architecture with N_m master devices, N_s slave devices and one tank per side can be modeled as:

$$\begin{cases} \Lambda_m \ddot{x}_m + \mu_m \dot{x}_m + D_m \dot{x}_m = \omega_m x_{t_m} + F^{ext}_m \\ \dot{x}_{t_m} = \dfrac{\sigma_m}{x_{t_m}} \dot{x}^T_m D_m \dot{x}_m + \dfrac{1}{x_{t_m}} (\sigma_m P^{in}_m - P^{out}_m) - \omega^T_m \dot{x}_m \\ \Lambda_s \ddot{x}_s + \mu_s \dot{x}_s + D_s \dot{x}_s = \omega_s x_{t_s} + F^{ext}_s \\ \dot{x}_{t_s} = \dfrac{\sigma_s}{x_{t_s}} \dot{x}^T_s D_s \dot{x}_s + \dfrac{1}{x_{t_s}} (\sigma_s P^{in}_s - P^{out}_s) - \omega^T_s \dot{x}_s \end{cases} \tag{2.34}$$

where $P^{in}_m, P^{in}_s \geq 0$ and $P^{out}_m, P^{out}_s \geq 0$ are incoming and outgoing power flows that the tanks can exchange with each other by means of the communication channel.

The policy used to define P_m^{out} and P_s^{out} in (2.34) is the same reported in [35] and is:

$$\begin{cases} P_m^{out}(t) &= (1-\sigma_m)(\dot{x}_m^T D_m \dot{x}_m - x_{t_m}\omega_m^T \dot{x}_m) + E_s^{req}(t-\delta)\beta_m \bar{P} \\ P_s^{out}(t) &= (1-\sigma_s)(\dot{x}_s^T D_s \dot{x}_s - x_{t_s}\omega_s^T \dot{x}_s) + E_m^{req}(t-\delta)\beta_s \bar{P} \end{cases} \tag{2.35}$$

where $\bar{P} \in \mathbb{R}_0^+$ is a design parameter that represents a rate of energy flowing from one tank to the other, and the flags E_m^{req}, E_s^{req} are used to implement an energy request process, and are defined as:

$$E_w^{req} = \begin{cases} 1, & \text{if} T_w(x_{t_w}) \leq T_w^{req} \\ 0, & \text{otherwise} \end{cases} \tag{2.36}$$

If the energy stored in the tank is under the user-defined threshold $T_w^{req} \in \mathbb{R}$ then the tank send an energy request signal E_w^{req} to the other tank, which can provide energy to the other side under the following condition:

$$\beta_w = \begin{cases} 1, & \text{if } T_w(x_{t_w}) \geq T_w^{ava} \\ 0, & \text{otherwise} \end{cases} \tag{2.37}$$

namely, each tank can provide energy to the other side if the energy stored is over the user-defined threshold $T_w^{ava} \in \mathbb{R}$.

In presence of time delay Δt between the two sides:

$$\begin{cases} P_s^{in}(t) = P_m^{out}(t-\Delta t) \\ P_m^{in}(t) = P_s^{out}(t-\Delta t) \end{cases} \tag{2.38}$$

While the power is traveling from one side to the other, it is stored in the communication channel that becomes an energy storing element in the teleoperation system. In particular, as shown in [35]:

$$H_{ch}(t) = \int_{t-\Delta t}^{t} P_m^{out}(\tau) + P_s^{out}(\tau) d\tau \tag{2.39}$$

where $H_{ch}(t)$ is the energy stored in the communication channel.

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

Proposition 2. *The system in* (2.34) *is passive with respect to the pair* $((F_{m_1}^{ext}, \ldots, F_{m_{N_m}}^{ext}, F_{s_1}^{ext}, \ldots, F_{s_{N_s}}^{ext}), (\dot{x}_{m_1}, \ldots, \dot{x}_{m_{N_m}}, \dot{x}_{s_1}, \ldots, \dot{x}_{s_{N_s}}))$.

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

$$W(t) = V_m(t) + V_s(t) + T_m(t) + T_s(t) + H_{ch}(t) \tag{2.40}$$

Using (2.34) it follows that:

$$\begin{aligned}\dot{W}(t) = \dot{x}_m^T F_m^{ext} - \dot{x}_m^T D_m \dot{x}_m + \dot{x}_s^T F_s^{ext} - \dot{x}_s^T D_s \dot{x}_s + \sigma_m(\dot{x}_m^T D_m \dot{x}_m) + \sigma_m P_m^{in}(t) + \\ - P_m^{out}(t) + \sigma_s(\dot{x}_s^T D_s \dot{x}_s) + \sigma_s P_s^{in}(t) - P_s^{out}(t) + \dot{H}_{ch}\end{aligned} \tag{2.41}$$

From (2.39):

$$\dot{H}_{ch}(t) = P_m^{out}(t) - P_m^{out}(t - \Delta t) + P_s^{out}(t) - P_s^{out}(t - \Delta t) \tag{2.42}$$

Considering (2.38) and replacing (2.39) in (2.41) it follows that:

$$\begin{aligned}\dot{W}(t) = \dot{x}_m^T F_m^{ext} + \dot{x}_s^T F_s^{ext} - (1 - \sigma_m)\dot{x}_m^T D_m \dot{x}_m - (1 - \sigma_s)\dot{x}_s^T D_s \dot{x}_s + \\ - (1 - \sigma_m) P_s^{out}(t - \Delta t) - (1 - \sigma_s) P_m^{out}(t - \Delta t)\end{aligned} \tag{2.43}$$

Since $\sigma_m, \sigma_s \in \{0, 1\}$ and $P_m^{out}(t - \Delta t), P_s^{out}(t - \Delta t) \geq 0$:

$$\dot{W}(t) \leq \dot{x}_m^T F_m^{ext} + \dot{x}_s^T F_s^{ext} \tag{2.44}$$

whence:

$$\dot{W}(t) \leq \sum_{j=1}^{N_m} \dot{x}_{m_j}^T F_{m_j}^{ext} + \sum_{z=1}^{N_s} \dot{x}_{s_z}^T F_{s_z}^{ext} \tag{2.45}$$

which implies the following passivity condition:

$$W(t) - W(0) \leq \int_0^t \left(\sum_{j=1}^{N_m} \dot{x}_{m_j}^T(\tau) F_{m_j}^{ext}(\tau) + \sum_{z=1}^{N_s} \dot{x}_{s_z}^T(\tau) F_{s_z}^{ext}(\tau) \right) d\tau \tag{2.46}$$

□

2.1.6 Validation

This section reports the validation of the architecture presented so far and is organized into two different experiments:

- experiment 1: the first experiment is performed by simulating a standard teleoperation session. This experiment shows the performances of the teleoperation system and prove the efffectiveness of the proposed architecture.
- experiment 2: the second experiment is performed by pushing manually one of the arm while moving the arm in the same direction. This experiment highlights the main advantages the proposed architecture reports with respect to the standard two-layer approach.

All the experiments are performed introducing a time delay $\delta t = 300ms$ between the master and the slave side in order to show the robustness of the control architecture.

A picture of the set-up used for this validation is reported in Figure 2.4: a KUKA LWR 4+ 7-DOF and a Universal Robots UR5 6-DOF endowed with 3D printed laparoscopic tools. At the master side, a prototype of the SARAS master console was used.

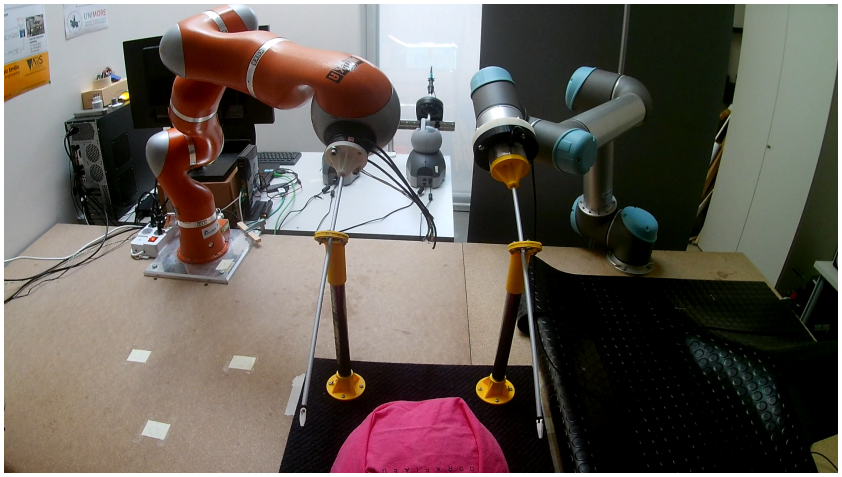


Figure 2.4: The first validation set-up.

The transparency layer has been modeled as follow:

$$\begin{cases} F^d_{m_j} = F_{ca_j} + F^{ext}_{s_j} \\ F^d_{s_z} = -K_p(x_{m_z} - x_{s_z}) - K_d(\dot{x}_{m_z} - \dot{x}_{s_z}) \end{cases} \tag{2.47}$$

where $F_{ca_j}(t) \in \mathbb{R}^n$ represents a force introduced for providing a feedback to avoid collisions between the slave devices, $F^{ext}_{s_j}(t) \in \mathbb{R}^n$ is the external force applied to the slave arm j that can be measured by using a force/torque sensor, K_p and K_d are the position error gain and the velocity error gain, respectively, and $j = 1, ..., N_m$ while $z = 1, ..., N_s$.

The collision avoidance forces $F_{ca_j}(t)$ have been introduced as virtual fixtures [58] to avoid collisions between the arms, which hare the same workspace. In the surgical scenario, the pivoting movement of the laparoscopic tools need to be always guaranteed. For this reason, collision avoidance is achieved by providing force feedback only at the master side, so that the motion of each slave arm remains constrained to the pivoting point by the mechanics of the master device. Collisions are avoided following three different steps:

- the tool of each slave arm, including the end-effector, is divided into c points around which a sphere of radius r_i, with $i = 1, ..., c$, is virtually built.
- the force generated by the contact of each sphere of the i-th slave arm and the j-th slave arm, with $j \neq i$, is computed.
- the overall force to feedback to the user is then computed considering all the forces and the application points and the levers effect on both master and slave sides.

The variable damping function was set as:

$$\xi(T_w(t)) = D^{min}_w + (D^{max}_w - D^{min}_w)\big(6(1-\lambda)^5 - 15(1-\lambda)^4 + 10(1-\lambda)^3\big) \tag{2.48}$$

with:

$$\lambda = \frac{T_w(t) - T^{b_{min}}_w}{T^{b_{max}}_w - T^{b_{min}}_w} \tag{2.49}$$

for both the $w \in \{m, s\}$ sides, in order to implement a smooth transition between D^{min}_w and D^{max}_w, avoiding discontinuities in the forces applied to the devices.

Table 2.1 shows the main controller parameters used for the experimental evaluation.

Parameter	Master side	Slave side	Unit
T_w^{max}	2.0	2.0	J
T_w^{min}	0.01	0.01	J
$T_w^{b_{max}}$	1.0	1.0	J
$T_w^{b_{min}}$	0.5	0.5	J
T_w^R	0.7	0.7	J
T_w^{ava}	1.0	1.0	J
T_w^{req}	0.7	0.7	J
$\bar{P}$	0.05	0.05	J

Table 2.1: Controller parameters used for the experimental evaluation.

Experiment 1

In the first part of the experiment, the laparoscopic tools on the slave robots were teleoperated in order to interact with a soft material, replicating a simplified interaction with human tissue. Then, the slave arms were moved trying to cause collisions, showing the action of the collision avoidance algorithm. For the sake of clarity and since the DOFs are usually chosen to be decoupled, all the plots report only one translational DOFs.

Figure 2.5-a reports the overall desired force and the real one. The desired force is given by the sum of the measured force at the slave side and the collision avoidance action, properly translated to fit with the master device. The mismatch between the desired force and the real one is due to the fact that the real force depends on the energy stored into the tank and to the damping counterpart introduced by the tank.

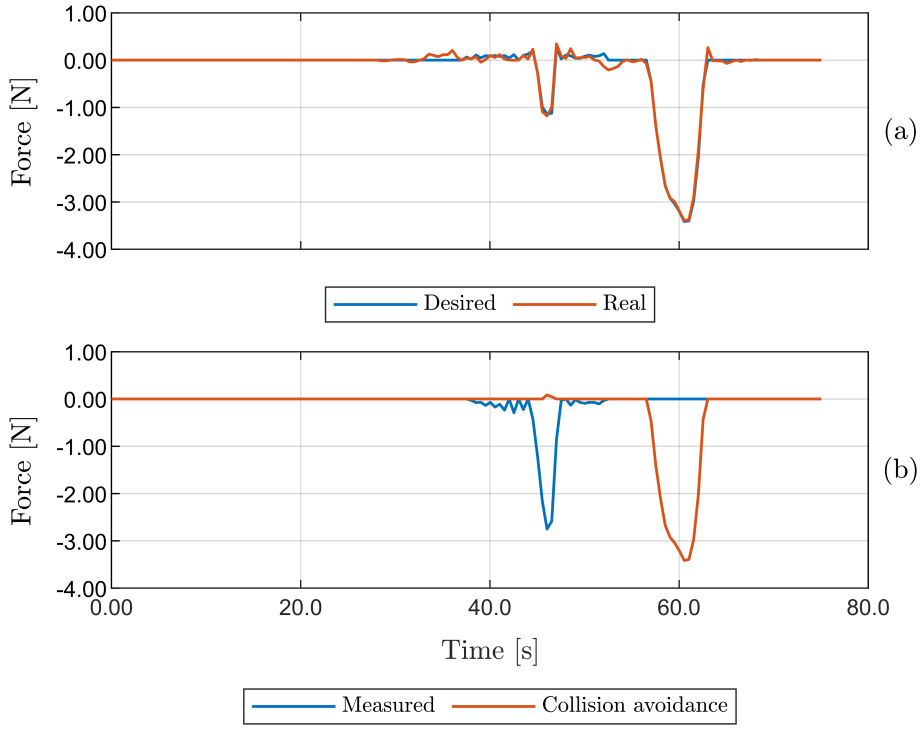


Figure 2.5: Experiment 1 - Forces exchanged on the master-left side.

Figure 2.5-b shows the interaction forces measured by the force/torque sensor, which reports a visible variation during the interaction, and the effect of the collision avoidance, which acts in the ending part of the experiment to guide the user to move the robots away, in order to avoid the collision.

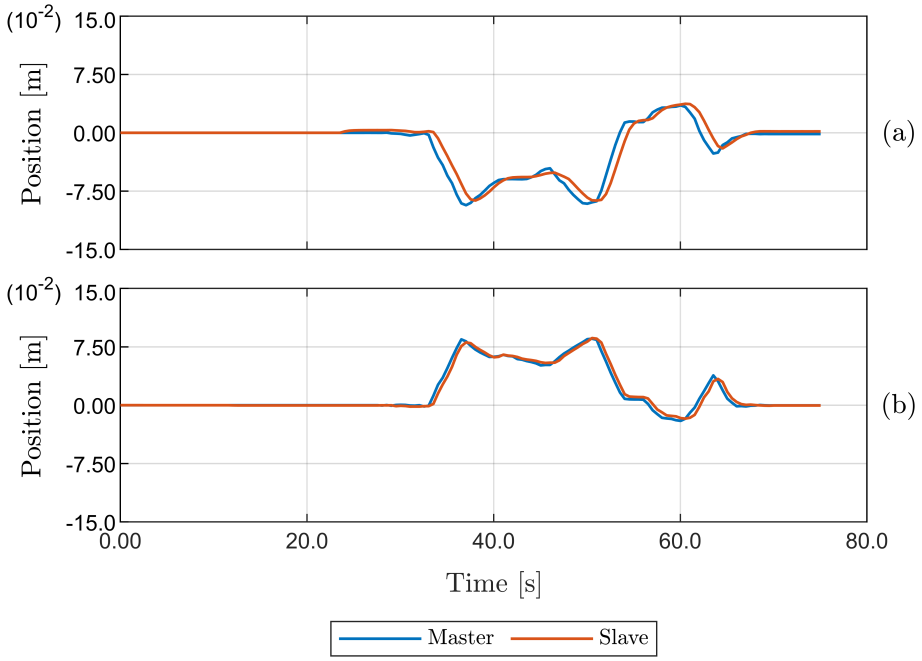


Figure 2.6: Experiment 1 - Cartesian position of the master and of the slave devices. (a) Right channel. (b) Left channel.

Figure 2.6 shows the Cartesian position of the master and of the slave devices. The communication delay introduced in the control architecture causes an evolution of the tracking error characterized by a maximum absolute value of $0.032m$ for the left side and of $0.014m$ for the right side. However, the average value turns out to be $0.0092m$ for the left side and $0.0046m$ for the right side, highlights a good behavior of the overall system.

Figure 2.7 shows the dissipated energy stored in the master and slave tanks. The evolution of the energy shows how the tanks are able to supervise the energy in the system, managing crucial conditions where unstable behaviors can occur, such as the interaction with the environment and the collision avoidance movement observable at time $45s$ and time $62s$, referable to the energy extraction of the tanks.

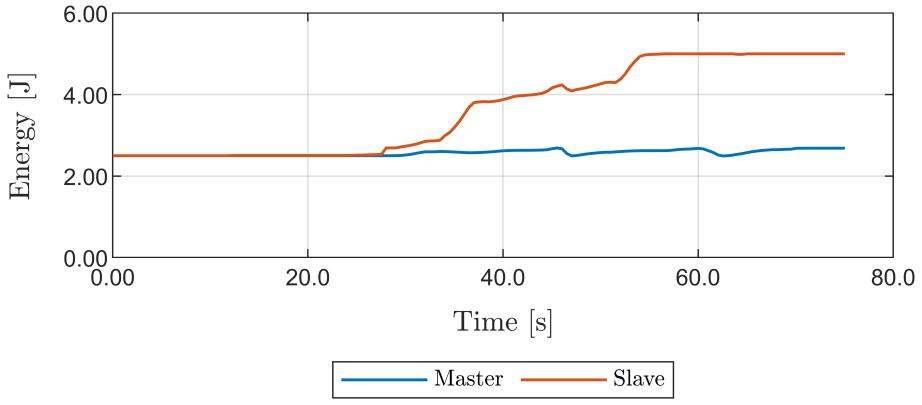


Figure 2.7: Experimet 1 - Energy stored in the master and slave tanks.

Experiment 2

In the second experiment, the teleoperation architecture was modified splitting the master shared energy tank and the slave shared energy tank into two independent master and slave energy tanks. The value of the initial energy and the damping coefficient introduced by the tank were also reduced.

During the experiment, an external force was applied to the right slave arm causing the emptying of the tank and stopping the robot, as can be seen in Figure 2.8-b, where the evolution of the energy stored in the tanks and the cartesian position of the master and slave devices are shown.

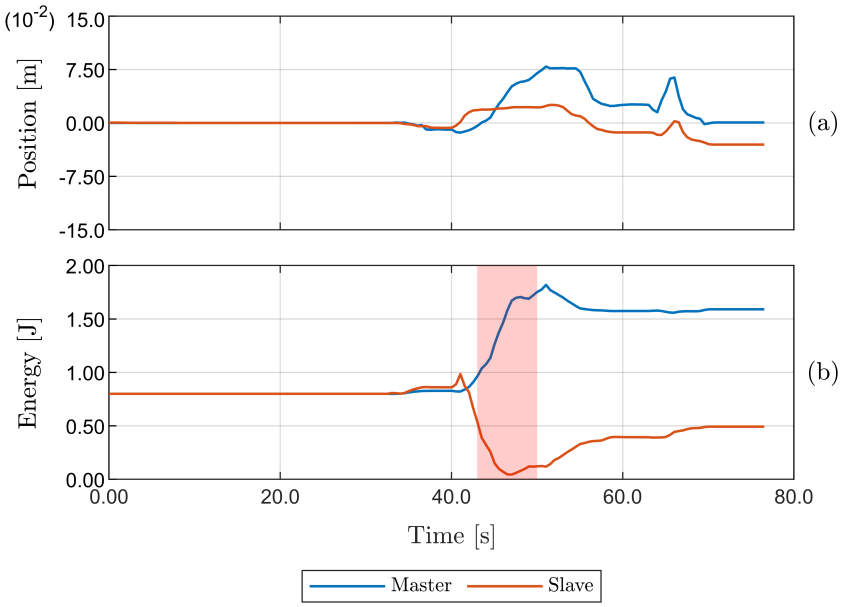


Figure 2.8: Experimet 2 - (a) Cartesian position of the master and of the slave right devices. (b) Energy stored in the master and slave right tanks.

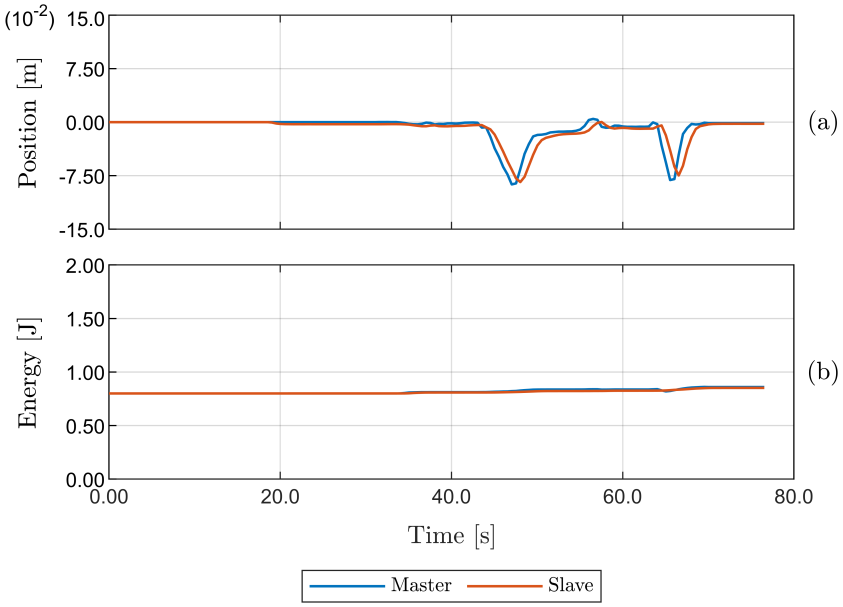


Figure 2.9: Experimet 2 - (a) Cartesian position of the master and of the slave left devices. (b) Energy stored in the master and slave left tanks.

At time $47s$ the energy stored in the right tank goes to its lower bound, inhibiting the movement of the robot, as highlighted by the shadowed red area in Figure 2.8-a. Looking at the cartesian position of the robot compared to the one commanded by the master side it is possible to notice that the arm stills in the same position while the master device moves, increasing the tracking error.

This behavior does not affect the left side, reported in Figure 2.9 which continues to work without issues. In such a case using the shared energy approach would have made the energy of the left robot available to the right robot, avoiding this kind of situation.

2.2 The SARAS Case Study

The teleoperation system for the SARAS MULTIROBOTS-SURGERY Platform described so far reported excellent results in terms of performance, but the first validation also highlighted some problems. Among these, in particular, inconsistencies and limitations related to how energy is used.

With the previous formulation of the shared energy tank (2.34) provided in Section 2.1.5, the upper bound of the energy stored in the tank generates some inconsistencies on how energy is used. In a single side system, this would not be a problem since the excess energy cannot be reused in order to practically maintain the passivity of the system, and the only way to bounding the energy is to waste it. Nevertheless, in a teleoperation scenario, the energy can be exchanged between the two sides without losing passivity. Therefore, a consistent management of the energy between the two sides is needed to reach the best performances.

This section reports all the improvements developed to address the aforementioned problem. In particular, the work focuses on the development of a new consistent formulation of the energy upper bounding strategy of the shared energy tank where the whole excess of energy can be reused decreasing the conservativeness of the overall system is developed.

2.2.1 Shared Energy Tanks

Consider the formulation of the shared energy tank in the teleoperations system as in (2.34):

$$\dot{x}_{t_w} = \frac{\sigma_w}{x_{t_w}} \dot{x}_w^T D_w \dot{x}_w + \frac{1}{x_{t_w}} (\sigma_w P_w^{in} - P_w^{out}) - \omega_w^T \dot{x}_w \tag{2.50}$$

where $w \in \{m, s\}$, where the subscripts m and s indicate the master and the slave side, respectively. The value of σ_w is define accordingly to (2.23) as:

$$\sigma_w = \begin{cases} 1 & \text{if } T(x_{t_w}) < T_w^{max} \\ 0 & \text{otherwise} \end{cases} \tag{2.51}$$

with, neglecting the energy request protocol:

$$P_w^{out}(t) = (1 - \sigma_w)(\dot{x}_w^T D_w \dot{x}_w - x_{t_w} \omega_w^T \dot{x}_w) \tag{2.52}$$

This means that when $T(x_{t_w}) = T_w^{max}$, namely the energy into the tank need to be bounded:

$$\sigma_w = 0 \tag{2.53}$$

hence:

$$P_w^{out}(t) = \dot{x}_w^T D_w \dot{x}_w - x_{t_w} \omega_w^T \dot{x}_w \tag{2.54}$$

and, therefore:

$$\dot{x}_{t_w} = -\frac{\dot{x}_w^T D_w \dot{x}_w}{x_{t_w}} \tag{2.55}$$

It is possible to observe that:

- the tank energy is decreased by an amount of energy proportional to the local damping, when this amount of energy should be sent to the other side of the teleoperation architecture.
- the energy related to the control action is sent to the other side of the teleoperation architecture regardless of its sign, i.e. a flow of energy from the environment to the tank (a dissipative action) or a flow of energy from the tank to the outside (active action).

These observations underline an inconsistency in the use of energy in this formulation of the tank.

To solve these problems, the new formulation of the shared energy tank is proposed as follow:

$$\dot{x}_{t_w} = \frac{\dot{x}_w^T D_w \dot{x}_w + \sigma_w P_w^{in}}{x_{t_w}} - \omega_w^T \dot{x}_w - \frac{P_w^{out}}{x_{t_w}} \tag{2.56}$$

with:

$$\sigma_w = \begin{cases} 0, & \text{if } T_w(x_{t_w}) = T_w^{max} \wedge \dfrac{\dot{x}_w^T D_w \dot{x}_w}{x_{t_w}} - \omega_w^T \dot{x}_m > 0 \\ 1, & \text{otherwise} \end{cases} \tag{2.57}$$

With this formulation, when $T(x_{t_w}) = T_w^{max}$, two scenarios arise:

1. the energy dissipated by the local damping is greater than the one requested by the control action. The overall flow of energy is positive and directed from the environment to the tank (dissipative action). The tank can no longer store energy, and this energy flow can be sent to the other side of the teleoperation architecture. Formally:

   $$\frac{\dot{x}_w^T D_w \dot{x}_w}{x_{t_w}} - \omega_w^T \dot{x}_m > 0 \tag{2.58}$$

   it follows that:

   $$\sigma_w = 0 \tag{2.59}$$

   and:

   $$P_w^{out}(t) = \dot{x}_w^T D_w \dot{x}_w - x_{t_w} \omega_w^T \dot{x}_w \tag{2.60}$$

   Therefore:

   $$\dot{x}_{t_w} = 0 \tag{2.61}$$

namely, the tank energy is kept constant at its maximum value and all the excess energy is sent to the other side of the teleoperation system by means of the port P_w^{out}.

2. the energy dissipated by the local damping is lower than the one requested by the control action. The overall flow of energy is negative and directed from the tank to the environment (active action). The tank is full and this energy must be extracted from it. Formally:

$$\frac{\dot{x}_w^T D_w \dot{x}_w}{x_{t_w}} - \omega_w^T \dot{x}_m < 0 \tag{2.62}$$

it follows that:

$$\sigma_w = 1 \tag{2.63}$$

and:

$$P_w^{out}(t) = 0 \tag{2.64}$$

Therefore:

$$\dot{x}_{t_w} = \frac{\dot{x}_w^T D_w \dot{x}_w}{x_{t_w}} - \omega_w^T \dot{x}_w \tag{2.65}$$

namely, the energy extracted from the tank is equal to the balance of the energy dissipated by the local damping and the one requested by the control action.

This means that with this new formulation it is possible to have a consistent flow of energy between both sides of the teleoperation system, providing also the correct bounding of the maximum energy stored into the tank. The following proves the stability of the overall system, with this new formulation.

2.2.2 The Bilateral Control Architecture

Formally, the overall architecture with N_m master devices, N_s slave devices, and one tank per side can be modeled, with the new formulation of the shared energy tank, as:

$$\begin{cases} \Lambda_m \ddot{x}_m + \mu_m \dot{x}_m + D_m \dot{x}_m = \omega_m x_{t_m} + F_m^{ext} \\ \dot{x}_{t_m} = \dfrac{\dot{x}_m^T D_m \dot{x}_m + \sigma_m P_m^{in}}{x_{t_m}} - \omega_m^T \dot{x}_m - \dfrac{P_m^{out}}{x_{t_m}} \\ \Lambda_s \ddot{x}_s + \mu_s \dot{x}_s + D_s \dot{x}_s = \omega_s x_{t_s} + F_s^{ext} \\ \dot{x}_{t_s} = \dfrac{\dot{x}_s^T D_s \dot{x}_s + \sigma_s P_s^{in}}{x_{t_s}} - \omega_s^T \dot{x}_s - \dfrac{P_s^{out}}{x_{t_s}} \end{cases} \tag{2.66}$$

All the considerations made in Section 2.1.5 regarding bilateral control architecture still hold.

Since the formulation of the tank changed, passivity need to be checked in order to guaranteeing the passivity of the system.

Lemma 1. *$P_w^{out}(t) \geq 0$ with $w \in \{m, s\}$.*

Proof. Since $\sigma_w, E_w^{req}, \beta_w \in \{0, 1\}$ and $\bar{P} \geq 0$, then:

$$E_w^{req}(t-\delta)\beta_w \bar{P} \geq 0 \tag{2.67}$$

Thus, from (2.35) it follows that $P_w^{out}(t)$ is positive if and only if:

$$(1-\sigma_w)(\dot{x}_w^T D_w \dot{x}_w - x_{t_w}\omega_w^T \dot{x}_w) \geq -E_w^{req}(t-\delta)\beta_w \bar{P} \tag{2.68}$$

If $\sigma_w = 1$, (2.68) becomes:

$$-E_w^{req}(t-\delta)\beta_w \bar{P} \leq 0 \tag{2.69}$$

that is always true thanks to (2.67). If $\sigma_w = 0$, from ((2.57)) it follows that:

$$(\dot{x}_w^T D_w \dot{x}_w - x_{t_w}\omega_w^T \dot{x}_w) \geq 0 \tag{2.70}$$

which satisfy (2.68). This is the only case where the input term contributes to $P_w^{out}(t)$. As a consequence, $P_w^{out} \geq 0$ and this proves the lemma. □

Proposition 3. *The MLMR teleoperation system in* (2.66) *is passive with respect to the pair ($(F_{m_1}^{ext}, \ldots, F_{m_{N_m}}^{ext}, F_{s_1}^{ext}, \ldots, F_{s_{N_s}}^{ext})$, $(\dot{x}_{m_1}, \ldots, \dot{x}_{m_{N_m}}, \dot{x}_{s_1}, \ldots, \dot{x}_{s_{N_s}})$).*

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

$$W(t) = V_m(t) + V_s(t) + T_m(t) + T_s(t) + H_{ch}(t) \tag{2.71}$$

Using (2.66) it follows that:

$$\begin{aligned}\dot{W}(t) = {} & \dot{x}_m^T F_m^{ext} - \dot{x}_l^T D_m \dot{x}_m + \dot{x}_m^T \omega_m x_{t_m} + \dot{x}_s^T F_s^{ext} - \dot{x}_r^T D_s \dot{x}_s + \dot{x}_s^T \omega_s x_{t_s} + \\ & + \dot{x}_l^T D_m \dot{x}_m - x_{t_m}\omega_m^T \dot{x}_m + \sigma_m P_m^{in}(t) - P_m^{out}(t) + \dot{x}_r^T D_s \dot{x}_s - x_{t_s}\omega_s^T \dot{x}_s + \\ & + \sigma_s P_s^{in}(t) - P_s^{out}(t) + \dot{H}_{ch}(t)\end{aligned} \tag{2.72}$$

While the power is traveling from one side to the other, it is stored in the communication channel that becomes an energy storing element in the teleoperation system. In particular:

$$H_{ch}(t) = \int_{t-\delta}^{t} P_m^{out}(\tau) + P_s^{out}(\tau) d\tau \tag{2.73}$$

From (2.73):

$$\dot{H}_{ch}(t) = P_m^{out}(t) - P_m^{out}(t-\delta) + P_s^{out}(t) - P_s^{out}(t-\delta) \tag{2.74}$$

and considering (2.38) and replacing (2.74) in (2.72) it follows that:

$$\dot{W}(t) = \dot{x}_l^T F_l^{ext} + \dot{x}_r^T F_r^{ext} - (1-\sigma_m)P_m^{in}(t) - (1-\sigma_s)P_s^{in}(t) \tag{2.75}$$

Since $\sigma_w \in \{0, 1\}$ and from Lemma 1 $P_w^{in} \geq 0$, it follows that:

$$\dot{W}(t) \leq \dot{x}_l^T F_l^{ext} + \dot{x}_r^T F_r^{ext} \tag{2.76}$$

whence:

$$\dot{W}(t) \leq \sum_{i=1}^{N_m} \dot{x}_{l_i}^T F_{l_i}^{ext} + \sum_{j=1}^{N_s} \dot{x}_{r_j}^T F_{r_j}^{ext} \tag{2.77}$$

which implies the passivity condition:

$$W(t) - W(0) \leq \int_0^t \sum_{i=1}^{N_m} \dot{x}_{l_i}^T(\tau) F_{l_i}^{ext}(\tau) + \sum_{j=1}^{N_s} \dot{x}_{r_j}^T(\tau) F_{r_j}^{ext}(\tau) d\tau \tag{2.78}$$

□

2.2.3 SARAS MULTIROBOTS Platform

The bilateral MMMS teleoperation architecture was developed for the SARAS MULTIROBOTS-SURGERY platform. With this platform has been also developed the SARAS Surgical Active System (which will hereafter be referred to as SARAS arm), and the SARAS master console.

In this platform, the da Vinci®surgical robot is used by the main surgeon, who performs the main procedure, while the assistant surgeon teleoperates the SARAS arms through the SARAS master console, helping the main surgeon during the entire procedure. This architecture allows performing the prostatectomy procedure on realistic phantoms, which faithfully reproduce the anatomy of the organs in the human pelvis.

In the particular case of SARAS, the teleoperation system is a bilateral DMDS architecture. However, the approach presented in the following is general and can be applied to any teleoperation system independently from the number of master and slave devices.

The SARAS Master Console

As shown in Figure 2.10, the SARAS master console is composed of four haptic devices: a pair of Simball joysticks (G-coder Systems AB, Britta, Sweden)[59] connected to a pair of Geomagic Touch 3D (3D Systems, Rock Hill, CA, USA)[60].

The use of the Simball joysticks as master devices for the SARAS master console allows to faithfully replicate the surgical scenario, giving the assistant surgeon the same tools used during real surgery. This allows the console to be extremely intuitive.

Since the Simball joysticks are not able to provide any kind of force feedback, the two Geomagic Touch 3D have been added to make the teleoperation system bilateral. Indeed, in this set-up, the two Simball joysticks faithfully return the position and orientation of the tip of the instrument, while the two Geomagic Touch 3D return the desired force feedback to the user.

The master console is also equipped with a 3D HMD, through which the assistant surgeon is provided with the view of the working area, augmented with useful information like, for example, forbidden regions.

Figure 2.10: SARAS Master devices

Figure 2.11 reports a user using the SARAS master console during the preliminary developments.

Figure 2.11: SARAS Master Console

The SARAS Arms

Figure 2.12 reports the CAD of one of the two SARAS Surgical Active System (see Section 1.2.3), developed specifically for the SARAS project: a mechatronic device holding and guiding a laparoscopic tool.

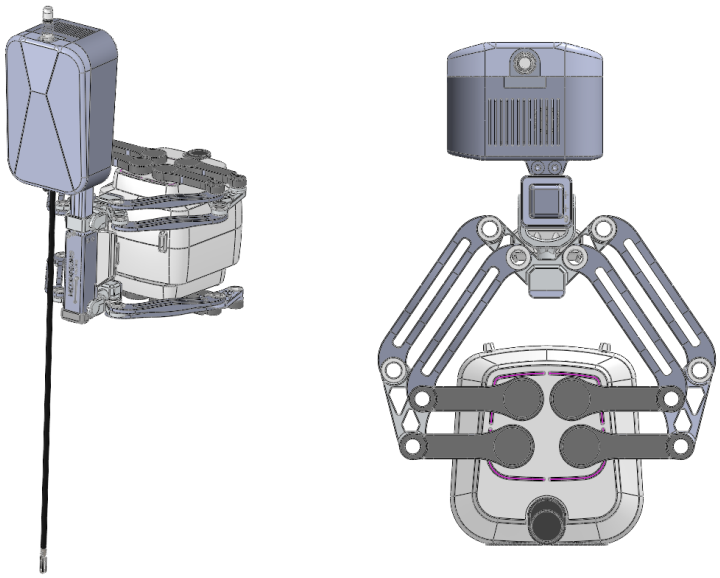


Figure 2.12: The CAD of the SARAS Surgical Active System

The kinematics and the linear axis provide 5 DOFs, which are reduced to 3 DOFs by the RCM constraints imposed by the software kinematic solver.

Three strain gauges are mounted 120 degrees apart on the shaft of the laparoscopic tools, as reported in Figure 2.13. The strain gauges measure deformations of the laparoscopic tool, which are directly converted into the directional forces. In this way, it is possible to feedback the interaction forces and supply them to the teleoperation system.

(a) a single strain gauge applied on the laparoscopic tool before being covered with a protective material.

(b) three strain gauges applied on the SARAS laparoscopic tool after being covered with a protective material.

Figure 2.13: The force sensor applied onto the shaft of the laparoscopic tool

2.2.4 Validation

This section reports the validation of the improved architecture, and is organized into three different experiments:

- experiment 1: the first experiment is performed moving freely the two SARAS arms, without colliding with obstacles. This experiment evaluates the transparency of the teleoperation architecture under the most favorable condition.
- experiment 2: the second experiment is performed moving the SARAS arms forward to the bladder in the manikin and starting to push it down several times. This experiment is intended to evaluate the stability of the system during interaction with the environment.
- experiment 3: the third experiment is performed keeping the SARAS arms stopped, and moving one of the da Vinci®arms to push one of the SARAS arms. This experiment tests the behavior of the platform when interacting with a non-passive system.

All the experiments are performed introducing a time delay $\delta t = 300ms$ between the master and the slave side in order to show the robustness of the control architecture.

A picture of the set-up used for this validation is reported in Figure 2.14: the two SARAS arms and the da Vinci®arms share the same workspace and operate together during a prostatectomy emulation. At the master side, the SARAS master console was used.

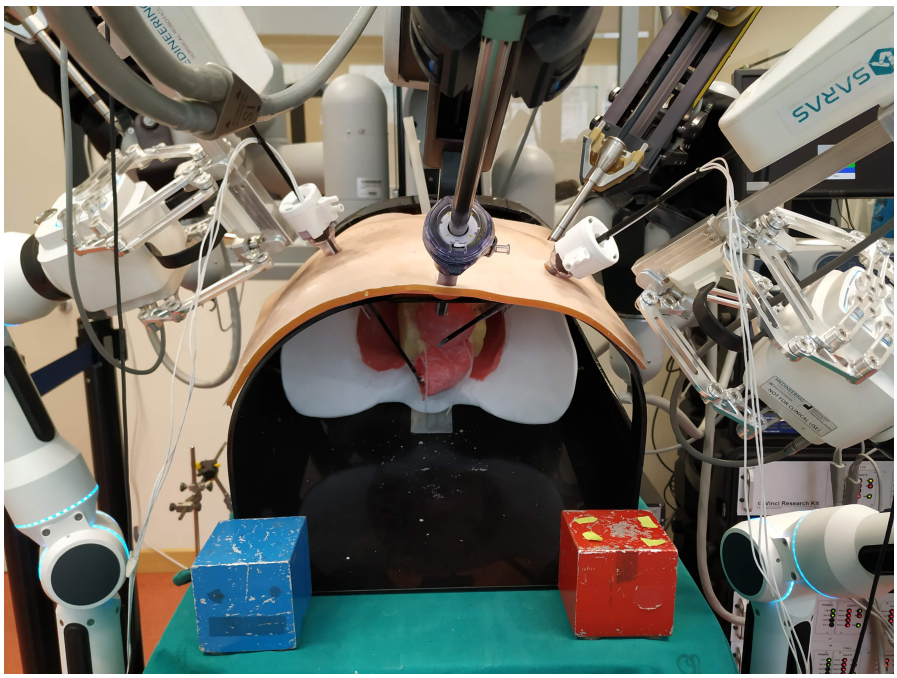


Figure 2.14: The SARAS set-up for prostatectomy emulation.

The SARAS arms robots controller provides information only about its position, and therefore it can be controlled only with position references. Since the passivity condition reported in Section 2.2.2 takes into account both the velocity and force of the robotics system, the proposed method can be used either with admittance or impedance causality.

This allows the architecture to provide a velocity reference, which integrated return the position reference for the SARAS arms. Consequently, the transparency layer is implemented as a force-velocity controller as:

$$\begin{cases} F^d_{m_i}(t) & = F^{ext}_{s_i}(t-\delta) \\ \dot{x}^d_{s_i}(t) & = \dot{x}_{m_i}(t-\delta) \end{cases} \quad i \in \{l, r\} \tag{2.79}$$

where the subscripts l and r are used to indicate the left and right channel, i.e. the left master and left slave, and the right master and right slave respectively.

Due to the limited workspace of the SARAS arms and to their relative positioning on the set-up, the collision avoidance force designed for the previous validation has been omitted, since the robots cannot physically collide except near to the tool-tip. Moreover, some operations performed by the surgeon require very low distances between the tools, for which it is not necessary to introduce repulsive forces.

The variable damping function has not been changed.

Table 2.2 shows the main controller parameters used for the experimental evaluation.

Parameter	Master side	Slave side	Unit
T_w^{max}	2.0	2.0	J
T_w^{min}	0.01	0.01	J
$T_w^{b_{max}}$	1.0	-	J
$T_w^{b_{min}}$	0.5	-	J
T_w^R	0.7	0.7	J
T_w^{ava}	1.0	1.0	J
T_w^{req}	0.7	0.7	J
$\bar{P}$	0.05	0.05	J

Table 2.2: Controller parameters used for the experimental evaluation.

Experiment 1

The results of the first experiment are reported in Figures 2.15, 2.16, 2.17, and 2.18.

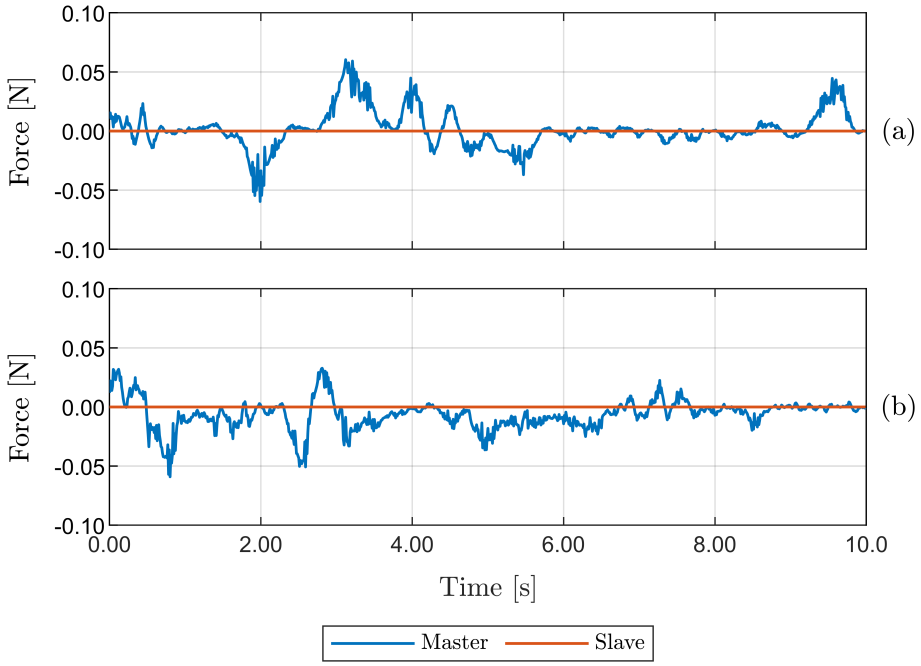


Figure 2.15: Experiment 1: Desired and commanded Cartesian force at the master device. (a) Left channel. (b) Right channel.

As visible in Figure 2.15, in this first experiment the interaction force measured at the slave side $F_{s_i}^{ext}(t)$ is equal to zero, as no interactions are carried out by the user. However, the forces applied at the master side $F_{m_i}^{\tau}(t)$ are not zero since the local damping is acting and modifying the applied forces $F_{m_i}^{\tau}(t)$ to dampen the master movements in relation to the velocities $\dot{x}_{m_i}$.

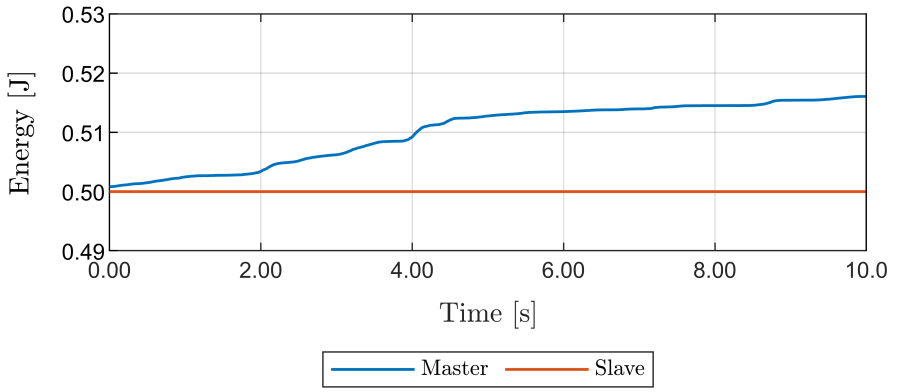


Figure 2.16: Experiment 1: Energy stored in the master and slave tanks.

As mentioned before, the SARAS arms can be controlled only with reference positions. This means that the local damping can be applied only at the master side, which harvests the energy for both sides of the teleoperation architecture. This behavior is notable in Figure 2.16 where the harvested energy makes the master energy T_m increase.

Since the energy threshold under which the damping value is set to its maximum value $T_m^{b_{min}}$ has been set equal to $0.7J$, and the initial energy stored in the master tank $T_m(0)$

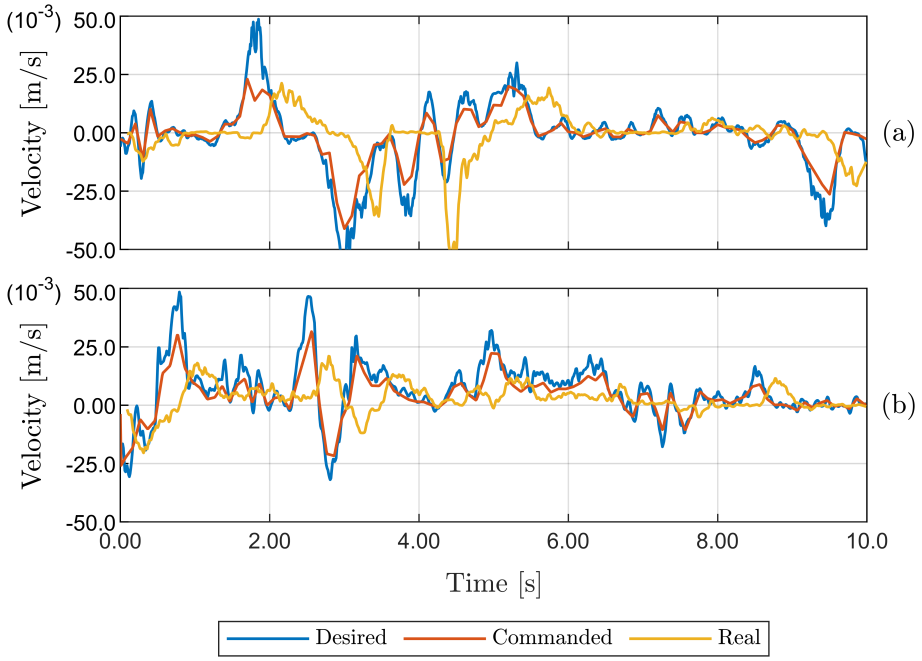


Figure 2.17: Experiment 1: Desired, commanded and real Cartesian velocity of the slave device. (a) Left channel. (b) Right channel.

is less than this value, it is possible to see the passifying effect of the passive layer on the velocities of the slave devices, reported in Figure 2.17. Indeed, the velocity of both the slave robots $\dot{x}_{s_i}(t)$ is always less, in module, than the desired value $\dot{x}^{\tau}_{m_i}(t)$.

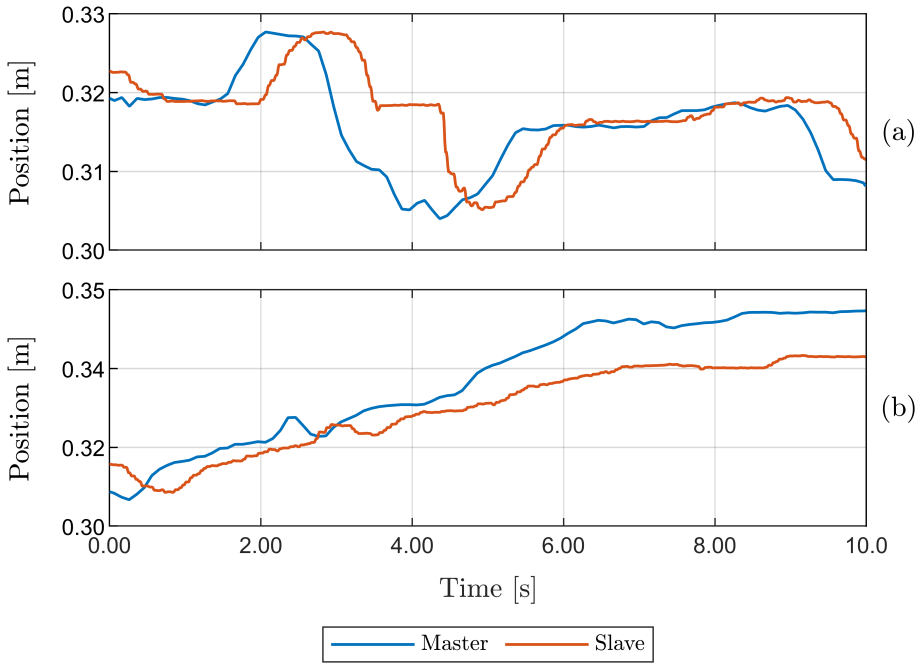


Figure 2.18: Experiment 1: Cartesian position of the master and of the slave device. (a) Left channel. (b) Right channel.

Figure 2.18 reports the Cartesian position of the master and of the slave devices. Considering: 1. the communication delay with round-trip-time of $0.6s$, 2. the computation of the desired position with the passifying action which decreases the nominal velocity, and 3. a maximum control frequency of 20Hz for the SARAS arms, the tracking performance can be considered good.

Experiment 2

The results of the second experiment are presented in Figures 2.19, 2.20, 2.21, and 2.22.

In this experiment, remarkable forces are measured at the slave side, as visible in Figure 2.19. The soft contact between the slave robots and the environment has a remarkable effect also on the energy dynamics. Indeed, the contact happens between $t \simeq 2s$ and $t \simeq 6s$, where the force measured at the left slave $F_{s_l}^{ext}$ is visibly greater than zero.

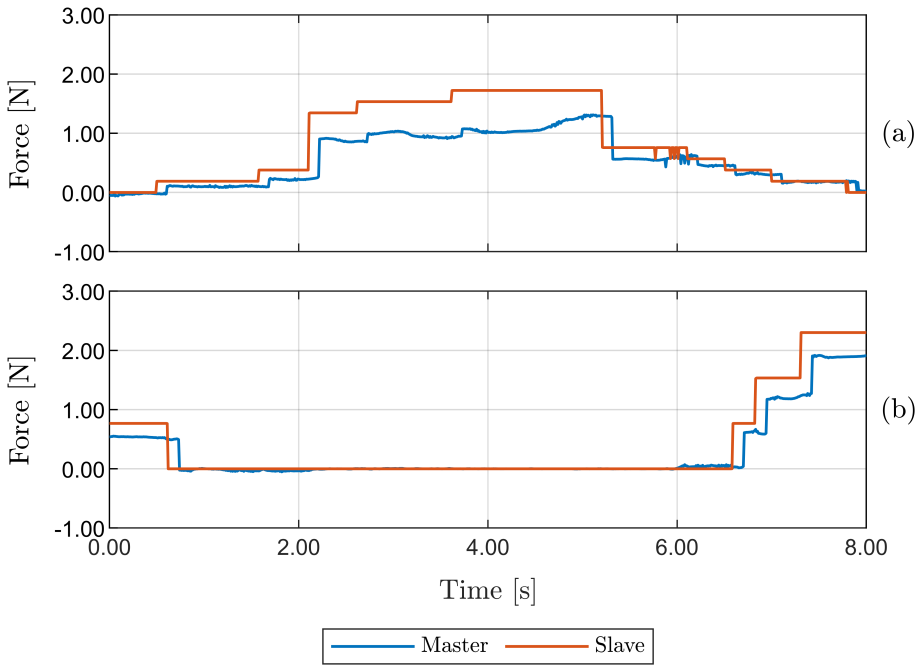


Figure 2.19: Experiment 2: Desired and commanded Cartesian force at the master device. (a) Left channel. (b) Right channel.

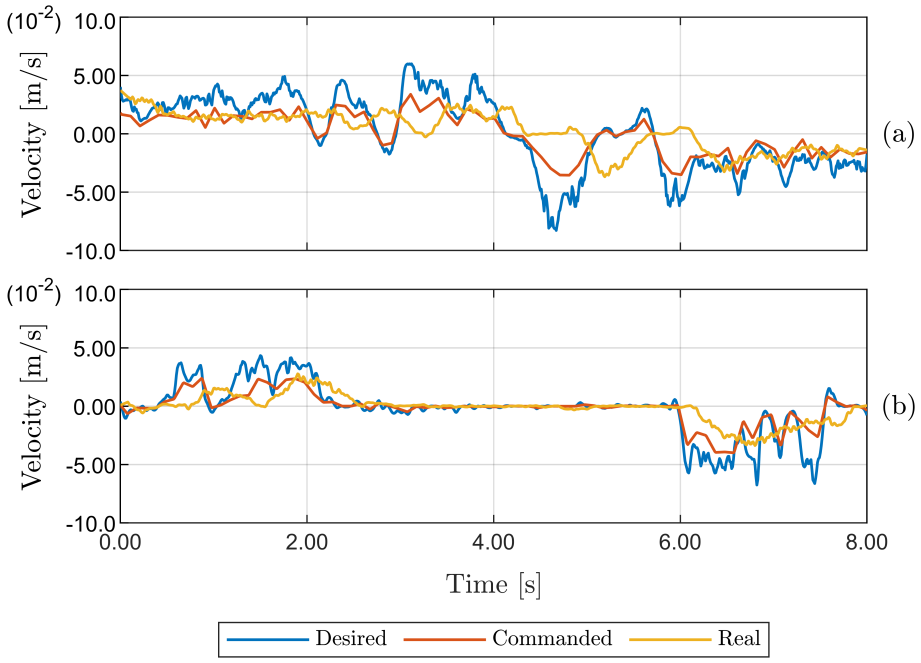


Figure 2.20: Experiment 2: Desired, commanded and real Cartesian velocity of the slave device. (a) Left channel. (b) Right channel.

In the same time interval, looking at Figure 2.20, the speed of the left slave $\dot{x}_{s_l}$ returns from a positive value, when the instrument moves from the rest position to the contact position, to a negative value, when the instrument moves towards the rest position after the contact.

This causes a first energy extraction from the slave tank when force and speed have the same sign, and a subsequent energy injection, when force and speed have opposite signs. This is visible in Figure 2.21.

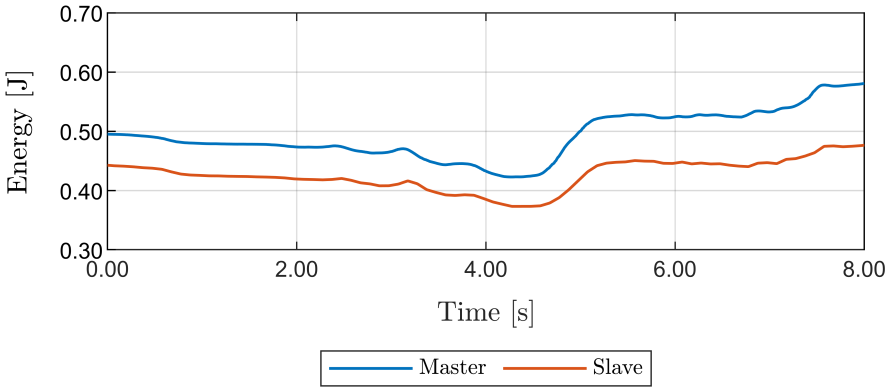


Figure 2.21: Experiment 2: Energy stored in the master and slave tanks.

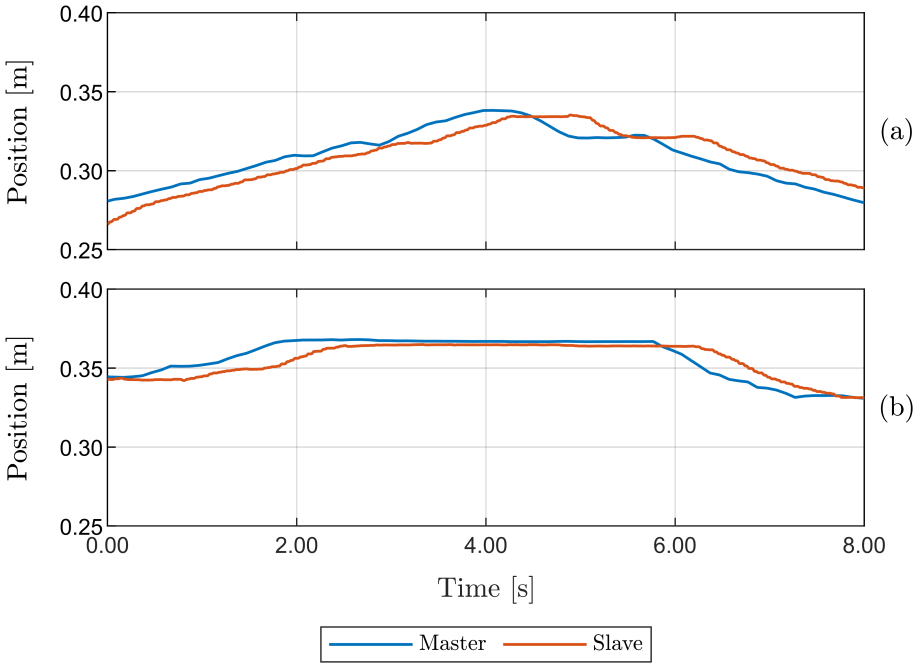


Figure 2.22: Experiment 2: Cartesian position of the master and of the slave device. (a) Left channel. (b) Right channel.

As in the previous experiment, $T_m^{b_{min}} = 0.7J$ and thus it is possible in Figure 2.20 to appreciate the passifying action of the passivity layer. The same consideration holds for the position tracking reported in Figure 2.22

Experiment 3

The experimental results of the third experiment are presented in Figures 2.23, 2.24, 2.25, and 2.26.

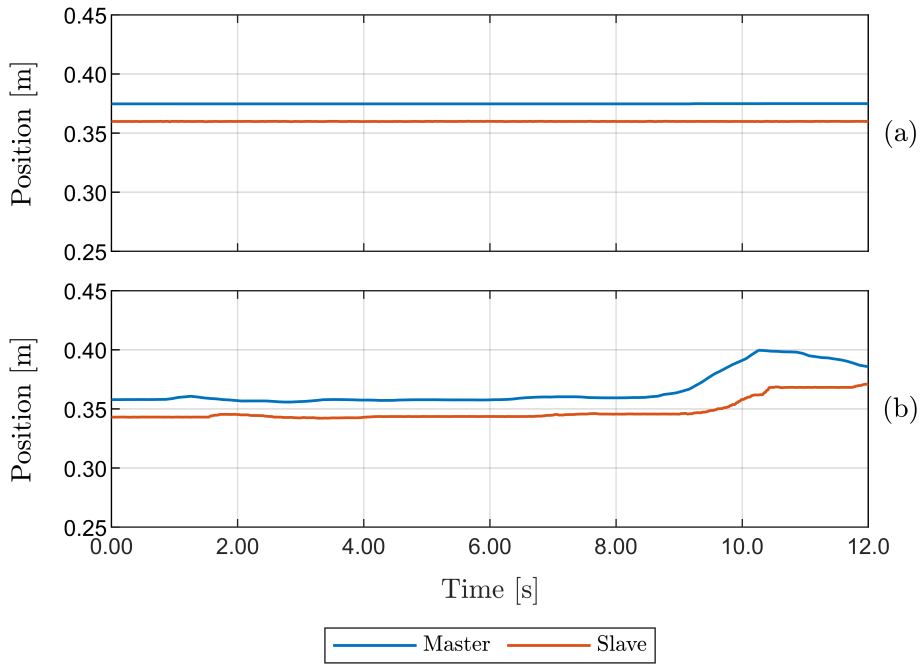


Figure 2.23: Experiment 3: Cartesian position of the master and of the slave device. (a) Left channel. (b) Right channel.

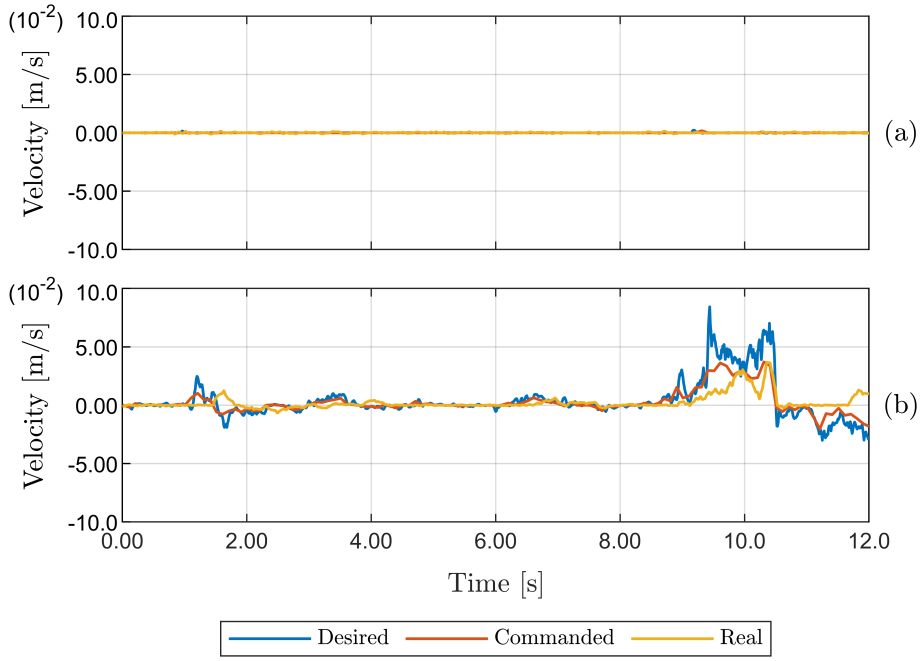


Figure 2.24: Experiment 3: Desired, commanded and real Cartesian velocity of the slave device. (a) Left channel. (b) Right channel.

In this experiment, the da Vinci®arm and the right slave robot get in contact at $t \simeq 10s$. This behavior can be observed looking at the positions of the right salve x_{s_r} in Figure 2.23, which are fixed, while the measured interaction forces $F_{s_r}^{ext}$ reported in Figure 2.25 are increasing.

When the interaction forces are at their maximum, the surgeon, which was handling the haptic tool with a relaxed grasping, loses the fixed position and follows the same direction of the interaction force. This behavior causes a substantial decrease in the energy level within the tanks, as visible in Figure 2.26. The action of the non-passive environment is well reflected in the energy dynamics of the proposed teleoperation system.

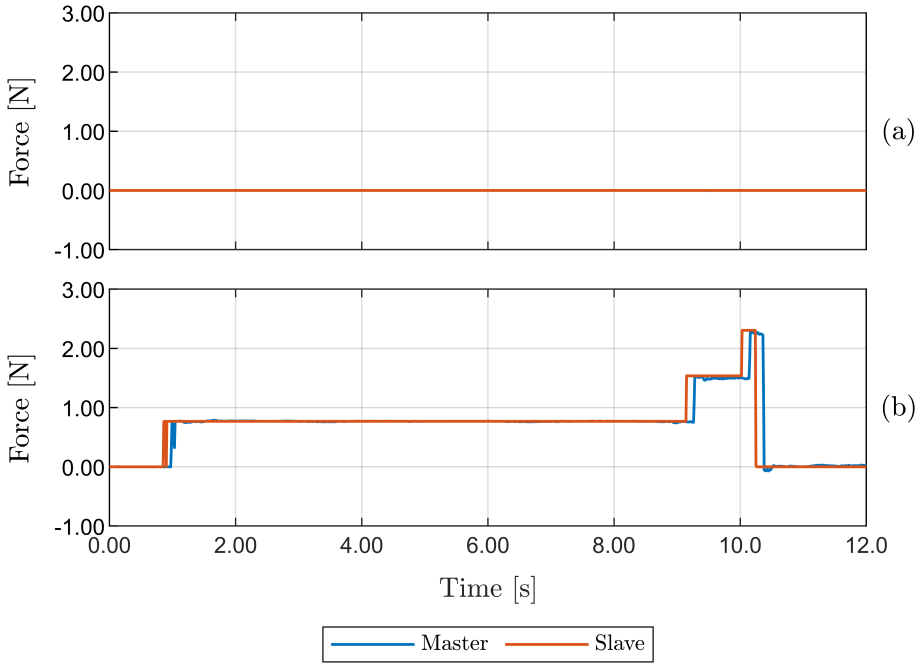


Figure 2.25: Experiment 3: Desired and commanded Cartesian force at the master device. (a) Left channel. (b) Right channel.

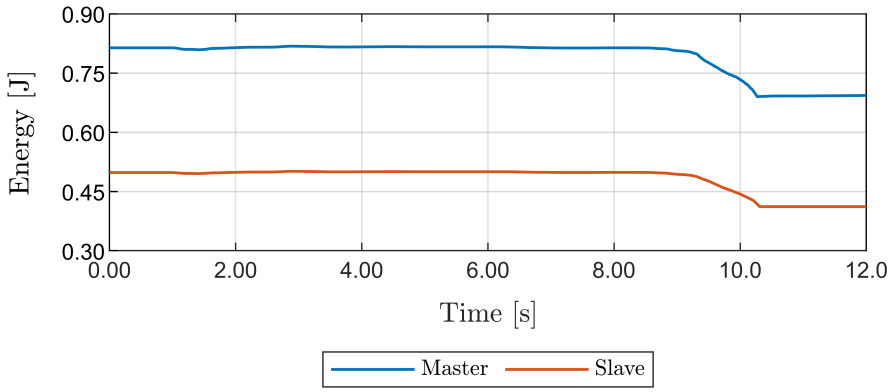


Figure 2.26: Experiment 3: Energy stored in the master and slave tanks.

2.3 Trilateral Teleoperation Architecture

2.3.1 Introduction

One of the main problems related to RAMIS is the training of novice surgeons [61]. Nowadays, the typical method by which a novice surgeon learns a new procedure is based on the traditional "see one, do one, teach one" method, and implies that the novice surgeon: *a*) attends a training course, *b*) watches the procedure performed by expert surgeons, *c*) performs his/her first operations with the guidance of a mentoring surgeon, and then *d*) performs the operations on his/her own. This mode of training is clearly inefficient and time-consuming.

Novel training systems and techniques that allow novice surgeons to learn while doing on the job can lead the way for more prevalent and effective use of teleoperation systems in surgical scenarios [62].

Teleoperation systems allow the user to remotely perform tasks that require high dexterity or movement scaling, feeling the interaction forces with the environment [38]. Several applications, including surgical training, require the presence of multiple users instead of a single operator to cooperatively control the slave movements. In that scenario, multilateral teleoperation systems provide the necessary level of flexibility and interaction capabilities. Examples of these applications in surgical training are [63] and [64]. The dual-console da Vinci Si Surgical System® (Intuitive Surgical, Inc., Sunnyvale, CA) released in 2009, represents another concrete example of Dual-Master-Single-Slave multilateral teleoperated system used for training purposes.

This work addresses the problem of controlling a trilateral teleoperation system in order to be able to implement a flexible, yet stable, interaction between the novice surgeon, the mentor and the surgical robot.

Figure 2.27 shows a schematic of that architecture. A surgical robot is placed in the operating room to perform the surgical procedure. A novice surgeon is seated on a remote console to teleoperate the robot. A mentor surgeon is seated on another remote console and he/she can take the control of the surgical procedure in order to show the novice surgeon how the procedure needs to be executed or to help him/her in critical situations.

Different control architectures have been developed for trilateral teleoperation systems. A common characteristic of these architectures is the presence of a dominance factor that gives authority over the execution of a task to one user rather than the other (see e.g. [64] [65]). The use of a dominance factor in a trilateral teleoperation system allows all the users to perceive all the actors in the network at the same time, making it difficult to understand if the feedback is coming from the user or from the environment.

The problem of stability has been addressed by exploiting the impedance model of the teleoperators and the environment [66] [67] or adaptive controllers [68] [69]. The main disadvantage of this kind of approaches is the lack of flexibility of the system. Indeed,

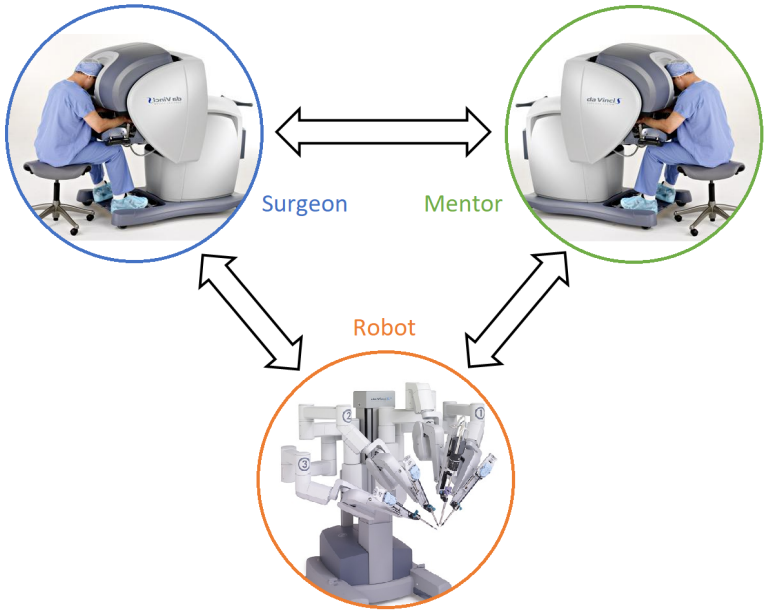


Figure 2.27: Schematic architecture of a trilateral teleoperation control system.

stability is guaranteed only when a specific interconnection between all the actors of the trilateral teleoperation system is specified. This prevents from changing online the relationship between the masters and slaves. In a mentoring scenario, this would mean having a constant mentoring relation, even when the novice surgeon is operating correctly.

Passivity-based approaches have been exploited for controlling MMMS teleoperation systems (see e.g. [70], [71],[55],[72]). While passivity is a more conservative property than stability, recent advances in passivity-based control (see e.g.[37],[73],[57]) allow reproducing flexible behaviors while ensuring passivity and, therefore, stability.

This work proposes a novel passivity-based trilateral teleoperation control architecture for the training of novice surgeons in a RAMIS surgical procedure. The training of the novice surgeon will be assisted by the remote guidance of a mentor. All the actors are communicating over a delayed communication channel. The control architecture has to ensure a stable and flexible behavior, namely the relations between the robotic systems can be time-varying, independently of the communication delay.

The two-layer approach [37, 35] is extended to a trilateral setting. At each side, the control architecture is split into two separate layers: the hierarchically higher layer is used to implement a strategy that addresses the desired transparency while the lower layer ensures that passivity is not violated, exploiting the concept of energy tank [74]. The proposed architecture allows extreme flexibility: a training strategy can be freely designed on the transparency layers and a stable implementation of the desired strategy is always guaranteed by the passivity layers.

This work proposes also a novel training strategy. The system can switch between two different states: the *manual state* and the *assisted state*. In the *manual state*, the novice surgeon teleoperates the robot executing the surgical task. In the *assisted* state is the mentor that teleoperates the robot executing the surgical task. In order to avoid confusing information, the force feedback due to the interaction of the robot with the environment

and the force feedback due to the interaction between the two masters are kept separated. The forces due to the interaction between the robot and the environment can be perceived only by the user who is teleoperating the robot. The user who does not teleoperate the robot feels a mismatch force, i.e. an elastic force that attracts her/his master position to the other user's master position. The switch between the two states can be managed only by the mentor. This choice allows the expert user, namely the mentor, to have greater authority over the execution of the task and to take control whenever a critical situation is detected.

The mismatch force allows *a*) the alignment between the masters to be maintained, *b*) the novice surgeon to understand and feel the mentor movements in order to learn how the task needs to be executed and *c*) the mentor to feel the novice surgeon movements in order to understand if it is necessary to take the lead of the operation.

The two-layer approach has been exploited in Section2 but, nevertheless, the considered scenario was made up of two sides exchanging energy, at each side, the robots were locally coupled through the tanks. Here three robotic systems and two users are interconnected over a communication channel and this complicates the stabilization and the definitions of the desired behavior of the robotic systems.

2.3.2 System Modeling

Consider a system composed of $N = 3$ robots, fully actuated and locally gravity compensated. Each robot, with n DOFs, can be modeled as (2.1) by considering $w \in \{s, m, r\}$ and the subscripts s, m, and r indicate the novice surgeon device, the mentor device, and the robot, respectively.

All the robots can exchange information over a delayed communication channel. For ease of presentation, the communication delay $\delta > 0$ is assumed to be constant. All the results can be easily extended to variable delays exploiting, e.g., the results of [75].

Each side of the teleoperation system is augmented with the tank (2.15) described in Section 2.1.4, which allows to model each side of the teleoperation system as (2.20).

As already demonstrated in Section 2.1.4, both the system in (2.1) and the augmented system (2.20) are passive with respect to the pair $(F_w^{ext}, \dot{x}_w)$.

2.3.3 The Trilateral Control Framework

The overall control architecture of the trilateral teleoperation system is reported in Figure 2.28.

For each component of the trilateral teleoperation system (novice surgeon, mentor and robot), two control layers can be recognized: a *Transparency Layer* and a *Passivity Layer*. As described in the previous section, each agent of the trilateral teleoperation system is augmented with a tank (*Tank surgeon*, *Tank mentor*, and *Tank robot* in Figure 2.28).

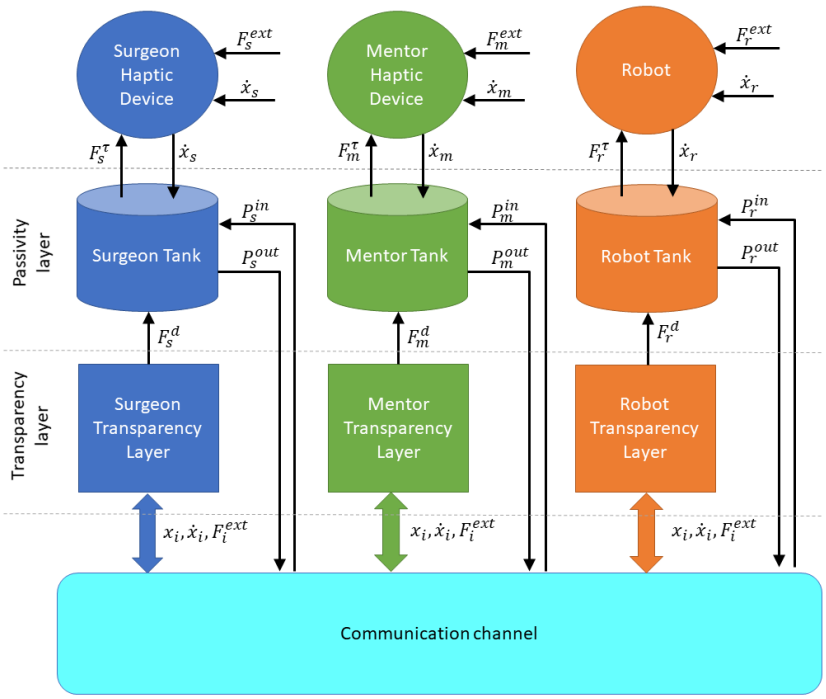


Figure 2.28: Coupling of the *surgeon haptic device*, the *mentor haptic device* and the *robot* by means of the communication channel.

In the Transparency Layer, each device exchanges position, velocity, and force information through the communication channel. This information is used to compute the desired inputs (F_s^d, F_m^d, F_r^d).

These forces are sent to the Passivity Layer, whose role is to passively implement them using the energy stored in the tanks.

As shown in (2.25), if the energy of a tank reaches a lower threshold, only a scaled version of the desired input can be implemented. The three tanks in the trilateral control architecture may store different levels of energy and, therefore, it may happen that one of the desired inputs has to be scaled while other tanks have more energy than the required one. Thus, in order to balance the energy in the tanks, the idea proposed in [35] is extended by allowing the three tanks to directly exchange energy. To this aim, each energy tank is endowed with a power input and a power output.

The overall system comprising the novice surgeon device, the mentor device and the robot can be thus modeled as:

$$\begin{cases} \Lambda_s \ddot{x}_s + \mu_s \dot{x}_s + D_s \dot{x}_s = \omega_s x_{t_s} + F_s^{ext} \\ \dot{x}_{t_s} = \dfrac{\sigma_s}{x_{t_s}} \dot{x}_s^T D_s \dot{x}_s + \dfrac{1}{x_{t_s}} (\sigma_s P_s^{in} - P_s^{out}) - \omega_s^T \dot{x}_s \\ \Lambda_m \ddot{x}_m + \mu_m \dot{x}_m + D_m \dot{x}_m = \omega_m x_{t_m} + F_m^{ext} \\ \dot{x}_{t_m} = \dfrac{\sigma_m}{x_{t_m}} \dot{x}_m^T D_m \dot{x}_m + \dfrac{1}{x_{t_m}} (\sigma_m P_m^{in} - P_m^{out}) - \omega_m^T \dot{x}_m \\ \Lambda_r \ddot{x}_r + \mu_r \dot{x}_r + D_r \dot{x}_r = \omega_r x_{t_r} + F_r^{ext} \\ \dot{x}_{t_r} = \dfrac{\sigma_r}{x_{t_r}} \dot{x}_r^T D_r \dot{x}_r + \dfrac{1}{x_{t_r}} (\sigma_r P_r^{in} - P_r^{out}) - \omega_r^T \dot{x}_r \end{cases} \tag{2.80}$$

where the tanks exchange power over the delayed communication channel according to:

$$P_w^{in}(t) = \sum_{i \in \{s,m,r\}, i \neq w} P_{i_w}^{out}(t - \delta) \tag{2.81}$$

namely the total input power $P_w^{in}(t)$ is equal to the sum of the output power coming from the other two tanks $P_{i_w}^{out}(t)$.

Following the approach proposed in [35], the outgoing power flows are designed as:

$$\begin{cases} P_{s_m}^{out}(t) = \rho(1-\sigma_s)\mathcal{D}_s/2 + E_m^{req}\beta_s\bar{P} \\ P_{s_r}^{out}(t) = (1-\rho/2)(1-\sigma_s)\mathcal{D}_s + E_r^{req}\beta_s\bar{P} \\ P_{m_s}^{out}(t) = (1-\rho)(1-\sigma_m)\mathcal{D}_m/2 + E_s^{req}\beta_m\bar{P} \\ P_{m_r}^{out}(t) = (1+\rho)/2(1-\sigma_m)\mathcal{D}_m + E_r^{req}\beta_m\bar{P} \\ P_{r_s}^{out}(t) = (1-\rho)(1-\sigma_r)\mathcal{D}_r + E_s^{req}\beta_r\bar{P} \\ P_{r_m}^{out}(t) = \rho(1-\sigma_r)\mathcal{D}_r + E_m^{req}\beta_r\bar{P} = P_{m_r}^{in}(t) \end{cases} \tag{2.82}$$

where $\mathcal{D}_w = \dot{x}_w^T D_w \dot{x}_w$ denotes the power dissipated by the w^{th} agent in the team and the parameter $\rho \in \{0, 1\}$ switches the behavior of the system from the *manual* state ($\rho = 0$) to the *assisted* state ($\rho = 1$) and it is set by the mentor.

The outgoing power $P_{w_i}^{out}$ is composed of the sum of two terms. The first term is given by the extra amount of dissipated power that cannot be stored in the w^{th} tank because the upper energy limit has been reached. The second term is given by a constant power flow that can be extracted by the tank.

The extra dissipated power is distributed among the agents according to their role. In fact, due to the different scale between the master devices and the robot, the tanks of the robot and of the user that is teleoperating it are the ones that need the biggest amount of energy for implementing the desired behavior [76]. Thus, using the strategy in (2.82), the tanks involved in the teleoperation of the robot exchange their extra amount of dissipated power while the third tank distributes equally its extra dissipated power to the other tanks.

If the energy stored is lower than T_w^R, the w^{th} tank sends an energy request signal E_w^{req} to the other tanks that enable the transmission of $\bar{P}$ to the w^{th} tank.

Considering the definition of the energy stored into the tank (2.16) with the coupling (2.18) with (2.80), returns:

$$\dot{T}_w = \sigma_w \dot{x}_w^T D_w \dot{x}_w + \sigma_w P_w^{in} - P_w^{out} + u_{t_w} y_{t_w} \tag{2.83}$$

that shows that the tank stores the dissipated power, the incoming power and it releases the outgoing power. The power exchanged through the power port (u_{t_w}, y_{t_w}) can be either stored in or released from the tank.

The overall trilateral control architecture is passive independently of the communication delay δ, as shown in the following:

Proposition 4. *The system composed of* (2.80)*, for* $w \in \{s, m, r\}$*, coupled according to* (2.81) *and* (2.82) *is passive with respect to the pair* ($(F_s^{ext}, F_m^{ext}, F_r^{ext})$, $(\dot{x}_s, \dot{x}_m, \dot{x}_r)$).

Proof. The power exchanged among the tanks, while traveling from one side to the other, is stored in the communication channel which, therefore, becomes an energy storing element. As evident from Figure 2.28, the power flowing in the communication channel is given by:

$$P_{ch}(t) = P^{out}(t) - P^{in}(t) \tag{2.84}$$

where P^{out} is the total power sent by the tanks. Denoting with ν the set $\{s, m, r\}$, the total power sent by the tanks P^{out} is given by:

$$P^{out}(t) = \sum_{w \in \nu} P_w^{out}(t) = \sum_{w \in \nu} \sum_{i \in \nu, i \neq w} P_{w_i}^{out}(t) \tag{2.85}$$

and P^{in} is the total power received by the tanks that, according to (2.81), is given by:

$$P^{in}(t) = \sum_{w \in \nu} P_w^{in}(t) = \sum_{w \in \nu} \sum_{i \in \nu, i \neq w} P_{i_w}^{out}(t - \delta) \tag{2.86}$$

By simple computations, starting from (2.85) and (2.86) it follows that $P^{in}(t) = P^{out}(t - \delta)$. Thus, the variation of the energy stored in the communication channel is given by:

$$\dot{H}_{ch}(t) = P^{out}(t) - P^{out}(t - \delta) \tag{2.87}$$

and, therefore, the energy stored in the communication channel is:

$$H_{ch}(t) = \int_{t-\delta}^{t} P^{out}(\tau) d\tau \tag{2.88}$$

From (2.82) it follows that $P^{out} \geq 0$ and that, therefore, $H_{ch}(t)$ is lower bounded.

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

$$W(t) = \sum_{w \in \nu} \Big(V_w(t) + T_w(t) \Big) + H_{ch}(t) \tag{2.89}$$

Considering (2.8),(2.80), and (2.83), following simple computations holds that:

$$\dot{W}(t) = \sum_{w \in \nu} \Big(\dot{x}_w^T F_w^{ext} - \dot{x}_w^T D_w \dot{x}_w + \sigma_w (\dot{x}_w^T D_w \dot{x}_w) + \sigma_w P_w^{in}(t) - P_w^{out}(t) \Big) + \dot{H}_{ch} \tag{2.90}$$

Considering (2.86), (2.85), and (2.87) in (2.90) it follow that:

$$\dot{W}(t) = \sum_{w \in \nu} \Big(\dot{x}_w^T F_w^{ext} + (1 - \sigma_w) \dot{x}_w^T D_w \dot{x}_w + \sigma_w P_w^{in}(t) - P_w^{out}(t - \delta) \Big) \tag{2.91}$$

Exploiting (2.81) and (2.82):

$$\dot{W}(t) = \sum_{w\in\nu}\Big(\dot{x}_w^T F_w^{ext} + (1-\sigma_w)\dot{x}_w^T D_w \dot{x}_w\Big) - \big[(1-\sigma_s)\big(P_{m_s}^{out}(t-\delta) + P_{r_s}^{out}(t-\delta)\big)\big] + \\ - \big[(1-\sigma_m)\big(P_{s_m}^{out}(t-\delta) + P_{r_m}^{out}(t-\delta)\big)\big] - \big[(1-\sigma_r)\big(P_{s_r}^{out}(t-\delta) + P_{m_r}^{out}(t-\delta)\big)\big] \tag{2.92}$$

Since $\sigma_s, \sigma_m, \sigma_r \in \{0,1\}$ and $P_{i_j}^{out}(t-\delta) \geq 0$ it follows that:

$$\dot{W}(t) \leq \sum_{w\in\nu} \dot{x}_w^T F_w^{ext} \tag{2.93}$$

which implies the following passivity condition:

$$W(t) - W(0) \leq \int_0^t \Big(\sum_{w\in\nu} \dot{x}_w^T(\tau) F_w^{ext}(\tau)\Big) d\tau \tag{2.94}$$

□

The previous result formally shows that the passivity of the trilateral teleoperation system which, therefore, is characterized by a stable behavior regardless of the implemented training strategy.

2.3.4 Validation

In order to evaluate the effective validity of the proposed architecture a single experiment in which a puncturing procedure needs to be executed is performed. In order to emulate a surgical training scenario, a beginner user and an expert user are placed at the two master consoles as if they were the novice surgeon and the mentor respectively. The experiment is performed forcing a time delay δ of 200ms between the three main actors in the architecture (namely the novice surgeon, the mentor, and the robot) in order to show the robustness of the proposed control architecture.

A pair of custom teleoperation devices composed of a 6 DOF Geomagic Touch haptic device coupled with a joystick with mechanical RCM are used as masters. At the slave side, a KUKA LWR 4+ 7-DOF robot endowed with 3D-printed laparoscopic tool is used to physically interact with the environment. To replicate the interaction with human tissues, the environment is composed of a rounded object made up of a soft material.

The transparency layer is set as:

$$\begin{cases} F_s^d = (1-\rho)F_r^{ext} + \rho\big(K_p^s e^{sm} + K_d^s \dot{e}^{sm}\big) \\ F_m^d = \rho F_r^{ext} + (1-\rho)\big(K_p^m e^{ms} + K_d^m \dot{e}^{ms}\big) \\ F_r^d = (1-\rho)\big(K_p^r e^{rs} + K_d^{rs} \dot{e}^{rs}\big) + \rho\big(K_p^r e^{rm} + K_d^{rm} \dot{e}^{rm}\big) \end{cases} \tag{2.95}$$

where K_p^g and K_d^g are the position error gains and the velocity error gains, with $g \in \{s,m,r\}$, and $e^{ij} = x_i(t) - x_j(t-\delta)$ with $i,j \in \{s,m,r\}$ are the position mismatch between the i-th device and the j-th device.

As said in Subsection 2.1.5, $\rho \in \{0, 1\}$ switches the behavior of the system from the *manual* state ($\rho = 0$) to the *assisted* state ($\rho = 1$). In the *manual* state, the novice surgeon teleoperates the robot and feels the environment while the mentor feels the mismatch force with the novice surgeon. In the *assisted* state is the mentor that teleoperates the robot feeling the environment while the novice surgeon feels the mismatch force with the mentor. The robot implements a simple PD controller exploiting the position error and the velocity error with the master of the user who is teleoperating the robot.

Table 2.3 reports the controller parameters used for the experimental evaluation.

Parameter	Value	Unit
T_w^{max}	5.0	J
T_w^{ava}	3.0	J
T_w^{R}	2.0	J
$T_w^{b_{max}}$	3.0	J
$T_w^{b_{min}}$	2.0	J
$\bar{P}$	0.05	J

Table 2.3: Controller parameters used for the experimental evaluation.

For hereafter, the user who teleoperates the robot will be referred to as the *active user* and the user who does not teleoperate the robot as the *passive user*.

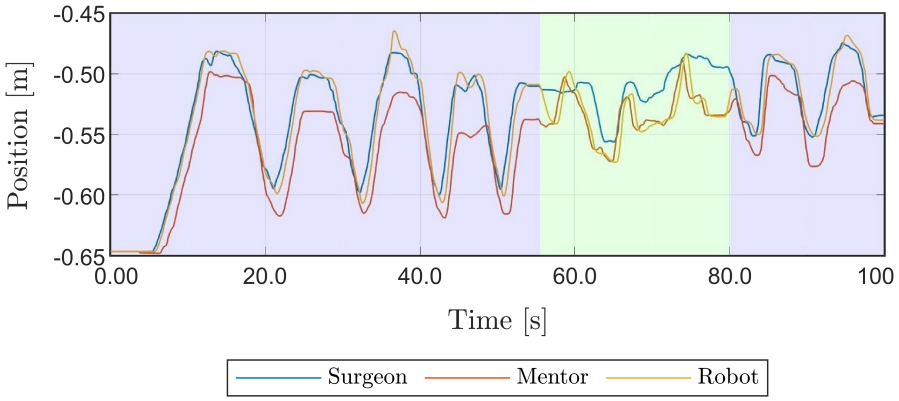


Figure 2.29: Cartesian position of the novice surgeon master device, of the mentor master device, and of the slave device (yellow line) along the x-axis.

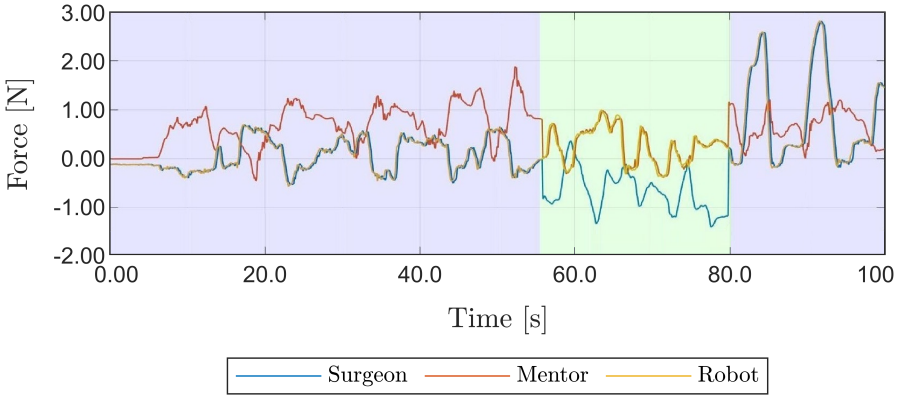


Figure 2.30: The force of the novice surgeon master device, of the mentor master device, and of the slave device along the x-axis.

Figure 2.29 reports the masters Cartesian position and the slave Cartesian position along the x-axis during the evolution of the experiment. For the sake of clarity and since the DOFs are usually chosen to be decoupled, the plots of only one translational DOF are reported. The blue and the green shadowed areas show when the *active user* is the novice surgeon or the mentor respectively. It can be clearly observed that the slave tracks the master position of the *active user*, namely, the novice surgeon in the blue shadowed area and the mentor in the green shadowed area. Furthermore, the position of the *passive user* is continuously attracted to the other. This can be stated by observing that the positions of the masters chase each other. This behavior is guaranteed by the implemented training strategy described in Section 2.3.1: the mismatch forces provided to the *passive user* try to maintain the alignment of the masters.

The forces returned to the masters and the force perceived at the slave side are depicted in Figure 2.30. As for the Cartesian positions, the forces perceived at the slave side are reflected to the *active user* while the *passive user* feels the mismatch forces. This allows to state the correct behavior of the system.

Figure 2.31 shows the evolution of the main quantities related to the energy tanks. The energy in the tanks evolves following the interaction with the environment, the damping injection, and the power exchange. In particular, can observed that the evolution over time of the *active user* energy is basically increasing while the evolution over time of the *passive user* energy is basically decreasing. For the *active user*, this is mainly due to the power flowing from the slave to the masters: the slave quickly reaches the maximum energy and sends excess power to the *active user*. In fact, $P_{r_s}^{out}$ is different from zero on the blue shadowed area and zero otherwise while $P_{r_m}^{out}$ is different from zero on the green shadowed area and zero otherwise. Exceptions occur only when an energy request is sent by one of the tanks, as observable at $t = 52s$ when the mentor requires energy. In that situation both the other tanks start to send energy as clear in the $P_{s_m}^{out}$ plot. For the *passive user* the energy evolution is basically decreasing since the effect of the mismatch forces is active, i.e. the movements and the forces have the same direction, causing an energy extraction.

2.4 Conclusions

This chapter deals with the development of bilateral teleoperation architectures capable of combining the use of multiple devices on both sides of the teleoperation system. The discussion of the proposed architecture is general, i.e. independent of the number of master and slave devices, and has been optimized to make energy management consistent and performant.

The first validation was necessary to highlight the weak points of the developed architecture, which were addressed in the second architecture. This second architecture was integrated within the SARAS MULTIROBOTS-SURGERY platform, making it possible to progress the project and to acquire the data necessary for the development of the next platform, the autonomous SARAS SOLO-SURGERY platform.

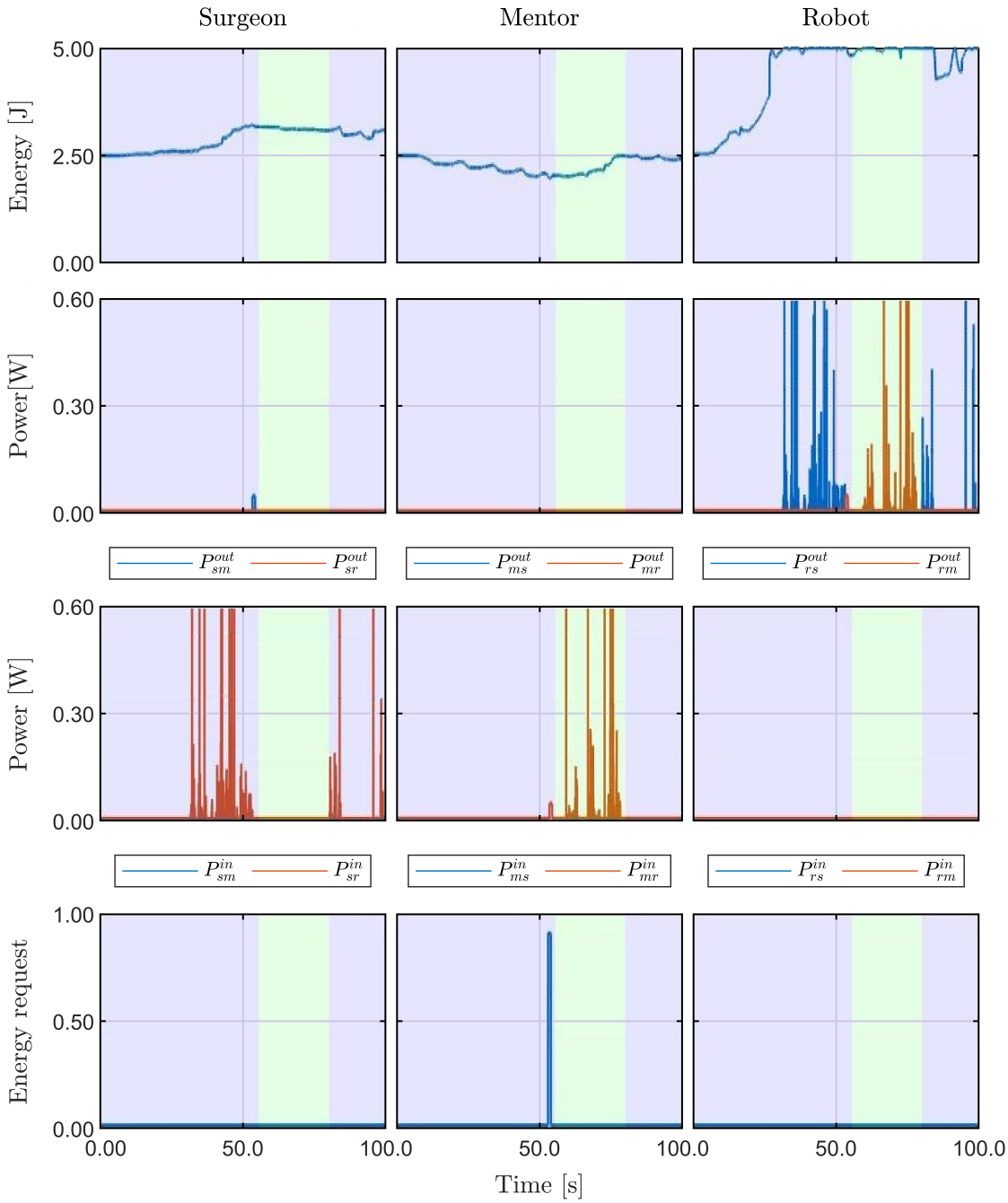


Figure 2.31: Main quantities related to the energy tanks. From above: energy store in the tank, output power flow, input power flow and energy request.

The final validation, as can be seen in Figure 2.14, was performed using realistic manikins and having real surgeons using the system by performing a prostatectomy procedure.

Using the same strategy, the teleoperation architecture has been extended and validated on a trilateral DMSS teleoperation architecture, proving the extreme flexibility of the proposed method.

In order to improve the capabilities of the system and quality of the surgical procedure during teleoperation, future work will aim at validating different types of controllers and multi-modal feedback.

Chapter 3

Model Predictive Control based Motion Planning for Surgical Scenario

This Chapter 3 reports the development of the Planning and Navigation module of the SARAS cognitive layer. The motion planning of multiple surgical tools operating in the same workspace is considered when constraints on the maximum velocity of the surgical tools and on the collisions between the tools need to be guaranteed. The problem related to the accuracy of the action recognition of the SARAS cognitive layer is also addressed including a confidence level in the constraints formulation. The results obtained after a first validation of the controller are extended in order to improve the performance of the system and to allow a real-time implementation. The controller is finally validated on a simulated setup and then experimentally on the SARAS setup.

The work presented in this chapter is published in [11, 12, 13, 4].

3.1 Introduction

Nowadays, developments in this field are increasingly aimed at introducing autonomy, and the medical community has high expectations in this direction. In [77] the image provided by the endoscope and a projected laser grid is used to identify the instruments and autonomously guide them towards the desired position. Autonomous suction of blood for hemostasis is proposed in [78] exploiting image-based blood flow detection. The system in [78] allows keeping clear the surgical field, which can be particularly occluded by the presence of blood.

Several techniques have also been developed for implementing automated suturing [79, 80, 81], while [82] reports a survey on robotized needle insertion.

Other works in terms of autonomy in surgical robotics are, for example, [83], where the authors proposed a cognitive control architecture designed to operate a surgical robot for needle insertion and suturing tasks in either teleoperated [35] and autonomous mode [84], guaranteeing a stable switch between the two and an adaptive interaction with the environment in both modes [35].

As already described in Section 1.1, during RAMIS the surgical tools are inserted into the patient's body through a small incision on his/her abdominal wall, which impose the RCM constraint to the surgical tools. Developing autonomous robots for RAMIS operations is further complicated, not only because of the motion constraint given by RCM, but also because such operations require the cooperation and coordination of several human actors.

To address this problem, a reactive approach to motion planning for a multi-arms laparoscopic surgical robot in a dynamic environment, based on MPC, is proposed. MPC-based approaches are considered a good choice in this scenario because they rapidly converge towards optimal solutions and allow to account for different types of constraints such as velocity limits and distance to obstacles [85]. Moreover, the presence of obstacles (*e.g.* tools and patient's organs) whose motion is predictable only within a short time horizon encourages the use of reactive methods like [86], and discourages the use of geometry-based and sampling-based planning algorithms.

Nevertheless, embedding the kinematic constraints of RAMIS in a MPC problem can be cumbersome. Therefore, a strategy for the computation of suitable waypoints is developed to guide the obstacle avoidance maneuver towards favorable directions, reducing the risk of being trapped into local minima.

The development of the overall architecture was split into two phases:

- a first architecture was developed with a non-linear MPC controller, through which it was possible to validate the effectiveness of the proposed control strategy.
- an improved and optimized architecture was later developed and validated, addressing the main problems arose with the previous architecture.

Both architectures were validated first in a simulated environment and, subsequently, experimentally on the SARAS set-up.

3.2 SARAS Case Study

The development of the Planning and Navigation module of the SARAS cognitive layer was built starting from the schematic architecture of the SARAS SOLO-SURGERY platform and of the SARAS LAPARO2.0-SURGERY platform reported in Figure 3.1 and Figure 3.2 respectively.

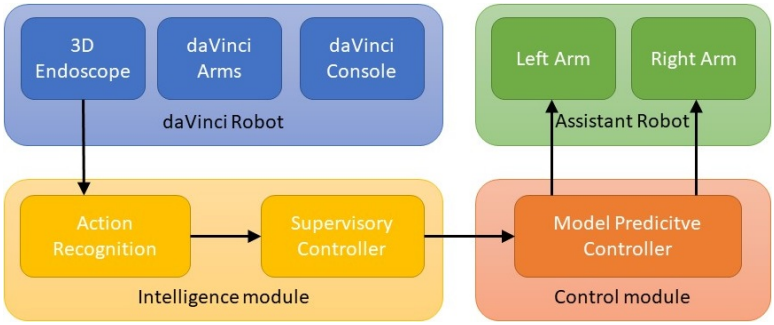


Figure 3.1: SARAS SOLO-SURGERY platform schematic architecture

In the SARAS SOLO-SURGERY platform, the main surgeon is teleoperating the da Vinci® tools while the assistant surgeon is substituted by the autonomous SARAS arms.

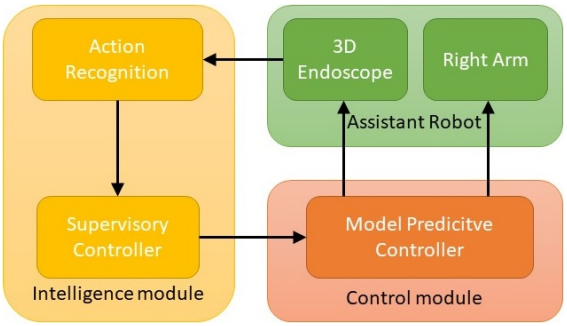


Figure 3.2: SARAS LAPARO2.0-SURGERY platform schematic architecture

In the SARAS LAPARO2.0-SURGERY platform, the main surgeon is operating using manual surgical tools while the assistant surgeon is still substituted by the autonomous SARAS arms.

Both architectures can be split into two sub-modules:

- the *Intelligence module*, that determines the desired behavior of the overall system.
- the *Control module*, that controls the robots in order to implement the desired behavior.

In particular, the *Action Recognition* sub-module exploits the endoscopic images to detect actions executed during the surgical procedure. The most likely action is then provided to the *Supervisory Control* sub-model, together with a confidence level associated with that detection. The progress of the surgical procedure is monitored by the *Supervisory Control* that computes the targets where the robots have to move, and sends them to the *Control module*, together with the confidence level. A *Model Predictive Controller (MPC)* is used to translate such targets into feasible trajectories for the arms.

From the *Control module* point of view, both platforms work in the same way: two laparoscopic tools are used by the main surgeon and two laparoscopic tools are controlled by the autonomous system. The main difference between the two platforms is that in the SARAS SOLO-SURGERY platform the main surgeon moves his/her instrument by the da Vinci®while in the SARAS LAPARO2.0-SURGERY platform he/she moves the tools manually. In both cases, the workspace of interest for collision avoidance is the internal part of the patient's abdomen.

From now on, the laparoscopic tools controlled by the autonomous system will be referred to as the *controlled tools*, and to the laparoscopic tools used by the main surgeonas as the *obstacle tools*. In the particular case of SARAS, the instrument involved are four. However, the approach presented in the following is general and can be applied to any system for laparoscopic surgery where tools guided by robotic arms are used.

The control system moves autonomously the *controlled tools* in order to reach the target configurations while dynamically avoiding collisions with the obstacles and between the *controlled tools*. The main control system is based on an MPC where the mathematical model of the two *controlled tools* is used to predict their future behavior and to compute the optimal input with which the robots can be controlled.

Based on an optimization process, the MPC can be affected by local minima, due to the presence of the *obstacle tools* in the workspace. To overcome this problem, the MPC is driven by computing intermediate waypoints on the basis of the current position of the *controlled tools*, the position of the *obstacle tools*, and the target position provided by the *Intelligence module*.

Compared to other solutions like [87, 88], this approach has the advantage of being specifically optimized for the guidance of laparoscopic tools in a shared environment with dynamical constraints. Therefore, operations can be performed in a tight environment by continuously searching for optimized solutions given both collision-free geometrical constraints over moving obstacles and varying velocities to respond to uncertainties.

3.3 Problem Statement

Consider a scenario in which the *controlled tools* are 4-DOFs velocity-controlled surgical robots holding laparoscopic tools. Velocity controlled robots are considered since this kind of robot is now widely used in many applications like the surgical one. Each robot can be modeled in the discrete-time domain as:

$$\bar{x}_j(k+1) = \bar{x}_j(k) + \bar{B}_j \bar{u}_j(k) \tag{3.1}$$

where $\bar{x}_j \in \mathbb{R}^3$ is the Cartesian linear position of the end-effector of each *controlled tool*, with $j \in \{r, l\}$ and the subscripts r and l used to represent the right and the left tool, respectively. The term $\bar{u}_j \in \mathbb{R}^3$ is the control input, i.e. the Cartesian linear velocity of the end-effector of each *controlled tool*, and the matrix $\bar{B}_j = \text{diag}\{\Delta t_c\} \in \mathbb{R}^{3\times3}$ is the input matrix, with Δt_c the sampling time ($t = k\Delta t_c$, $k \in \mathbb{Z}$).

The overall system considering both the *controlled tools* can be modeled as a single integrator in the discrete-time domain as:

$$x(k+1) = x(k) + Bu(k) \tag{3.2}$$

where $x = [\bar{x}_r, \bar{x}_l] \in \mathbb{R}^6$ is the state vector comprising the two tools, $B = \text{diag}\{\Delta t_c\} \in \mathbb{R}^{6\times 6}$ is the input matrix, and $u = [\bar{u}_r, \bar{u}_l] \in \mathbb{R}^6$ represents the control input.

The 4th degree of freedom of each *controlled tool*, i.e. the rotation along its main axis, is not included in the overall model (3.2) since it does not affect appreciably the obstacle avoidance, e.g. movements are performed with the tools tips closed or the tool is a needle.

The goal of the system is to move the *controlled tools* to the desired configuration avoiding collision with all the instruments in the workspace.

3.4 Robot-Obstacle Distance Computation

Since a safe interaction between the robotic systems and the patient needs always to be guaranteed, collisions between all the tools in the workspace have necessarily to be avoided.

Each tool in the workspace is modeled using virtual capsules, in order to enclose them into the fittest and simplest shape. This allows making the computation of the relative distance among the objects very simple and fast [89]. Given a pair of Cartesian points, a capsule is an object composed of two hemispheres, centered in these points, and a cylinder, with longitudinal axis linking the two points.

Let i and j be two generic tools into the workspace. The distance between the two capsules that enclose the i-th and j-th tool can be defined as:

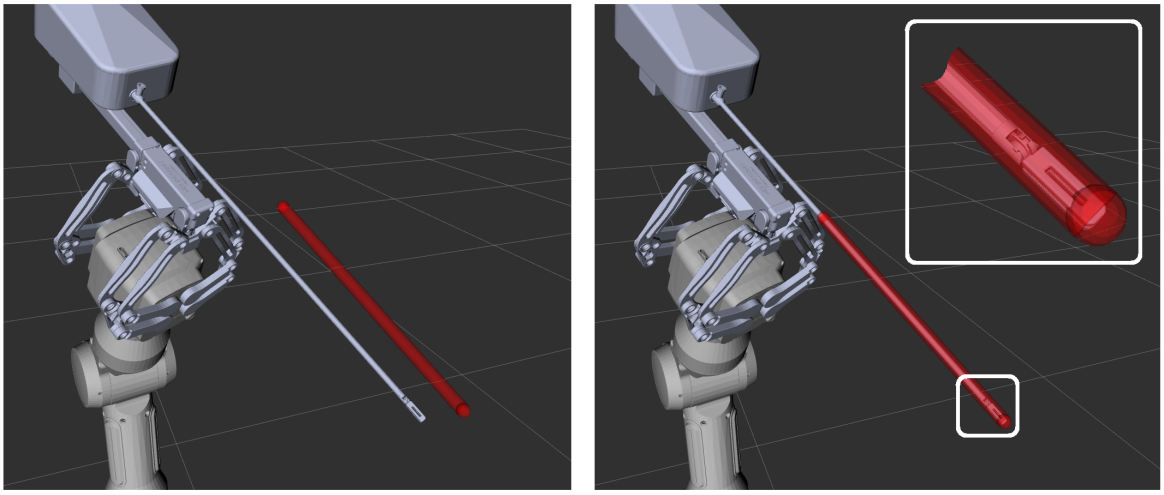
$$d_i^j = d_{ax_i}^{ax_j} - r_i - r_j \tag{3.3}$$

where $d_{ax_i}^{ax_j}$ is the distance between the axes of the capsules, computed as the distance between segments shown in [89], while r_i and r_j are the radii of the i-th and j-th tool virtual capsule.

Figure 3.3 shows the procedure used to build a capsule around a laparoscopic tool. Exploiting (3.3) the distances between the tools can be easily computed and embedded into the MPC to avoid collisions.

3.5 Design of the Model Predictive Controller

Safety-critical scenarios, of which surgical theaters are a prime example, require to enforce constraints on velocity limits and collision avoidance, that are taken into account to design this MPC-based control system.



(a) The CAD model of the robotic laparoscopic tool and, in red, a capsule.

(b) The capsule wrapping the robotic laparoscopic tool.

Figure 3.3: Procedure used to enclose a tool into a capsule.

At the same time, the control system has to deal with the uncertainty in the recognition of the action that comes from the Intelligence module. This uncertainty is the residual of the confidence level $\alpha \in [0, 1]$ paired to each action identified by the *Action Recognition* system within the *Intelligence module*.

The confidence level is used to modulate the Cartesian velocity of the robots while the control system is planning the motion of the robot towards the goal position. The intent is to avoid abrupt movements towards possibly incorrect goal positions, which are, on average, associated with low confidence levels, while avoiding stalling the system and impeding the motion of the teleoperated tools. As the confidence increases for correct predictions, the modulated tool velocity will consequently increase as well.

3.5.1 Constraints

Two types of constraints are considered in the MPC formulation:

- the velocity limit constraint.
- the collision avoidance constraint.

Velocity Limit Constraint

The velocities of the robots are physically limited. This can be translated into a bound on the control input:

$$||\bar{u}_j(k)|| \leq \alpha(k)\, \bar{u}_j^{\max} \tag{3.4}$$

where $\bar{u}_j^{\max} \in \mathbb{R}^+$ is the linear velocity limit and $\alpha(k)$ is the confidence level used to modulate the velocity of the robots depending on the uncertainty in the action recognition.

Collision Avoidance Constraint

As introduced in Section 3.4, collisions between the tools into the workspace need to be avoided during the task execution. To this aim, the distances between capsules computed as ((3.3)) are exploited and the collisions between the i-th tool and the j-th tool at time instant k are avoided by setting the following constraint:

$$d_i^j(k) \geq d_s \tag{3.5}$$

where $d_s \in \mathbb{R}$ is a user-defined positive parameter representing the safety distance.

3.5.2 Cost Function

The cost function of the MPC is defined as:

$$J(x^{\text{MPC}}, x) = \sum_{i=0}^{p-1} ||x^{\text{MPC}}(k) - \hat{x}(k+i)|| \tag{3.6}$$

where $p = N/\Delta t_c$ is the number of time steps in the prediction horizon N for the MPC sampling time Δt_c, $x^{\text{MPC}}(k) \in \mathbb{R}^6$ is the desired state at the time step k driven to the MPC, and $\hat{x}(k+i) \in \mathbb{R}^6$ is the predicted state with initial condition $\hat{x}(k+0) = x(k)$.

This cost function allows the *controlled tools* to reach the desired position with a straight trajectory if no *obstacle tools* are detected toward the goal. The desired state of the MPC $x^{\text{MPC}}(k)$ will be provided by the planner illustrated in Section 3.6.

The rotation and the velocity of the tool along its axis are controlled by an external proportional controller. This separation between linear and rotational control is possible by the nature of the 4-DOFs standard laparoscopy tools and the policy of moving the robots keeping the instruments closed.

3.5.3 MPC Formulation

The solution of the following constrained finite-horizon optimal control problem:

$$\begin{aligned}
\min_{\mathbf{u}} \quad & \sum_{i=0}^{p-1} ||x^{\text{MPC}}(k) - \hat{x}(k+i)|| \\
s.t. \quad & \hat{x}(k+i+1) = \hat{x}(k+i) + Bu(k+i) \\
& ||u_j(k)|| \leq \alpha(k)\, u_j^{\max} \\
& d_{r_r}^{r_l}(k+i) \geq d_s \\
& d_{r_j}^{o_h}(k+i) \geq d_s \\
& i = 0, ..., p-1 \\
& h = 0, ..., N_o \\
& j \in \{r, l\}
\end{aligned} \tag{3.7}$$

returns the optimal control input sequence $\mathbf{u} = [u(k), ..., u(k+p-1)]$.

In the optimization problem, $\hat{x}(k+i+1)$ represents the estimation of the state at time $k+i+1$ computed using the model ((3.2)), $u(k+i)$ is the linear control input at time $k+i$ and $d_{r_r}^{r_l}(k+i)$ is the distance between the virtual capsule built around the *controlled tools* at time $k+1$. $d_{r_j}^{o_h}(k+i)$ is the distance between the virtual capsules built around the j-th *controlled tools* and the h-th *obstacle tools* at time $k+1$ with N_o the number of *obstacle tools* in the workspace.

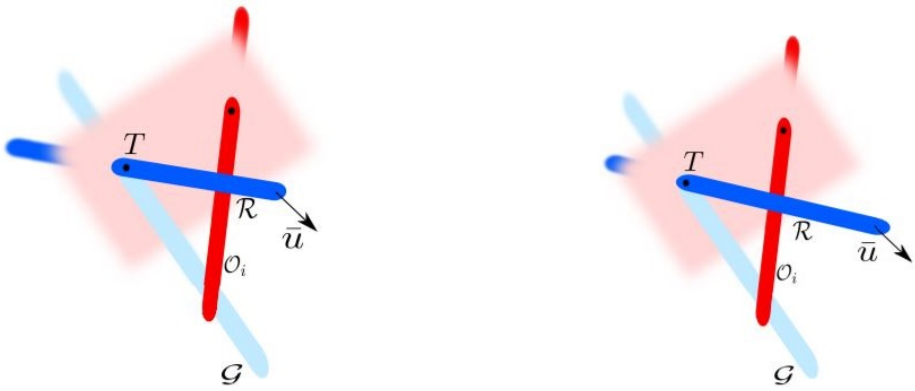
The position of each one of the N_o *obstacle tools* in the prediction horizon is computed considering its velocity to be constant over the entire horizon. The same holds for the desired position $x^{\text{MPC}}(k)$ and the confidence level $\alpha(k)$.

Finally, the first component $u(k)$ is used to compute the desired motion position $x^d(k+1) \triangleq x(k) + Bu(k)$ to be actually reached by the robots.

3.6 Waypoints Generation

If the target configuration, denoted as $x^g \in \mathbb{R}^3$, selected by the *Intelligence module* shown in Section 3.3 is directly provided as the MPC desired position $\bar{x}^{mpc}$, the presence of the *obstacle tools* inside the workspace does not guarantee that the use of the MPC alone may not be able to bring the tools to their desired position, even if all the MPC constraints are satisfies.

This problem is mainly caused to the RCM constraint. Consider the case of two generic tools as depicted in Figure 3.4a: a tool, represented by the blue capsule, has to reach its goal position, the light blue capsule, but an obstacle represented by the red capsule (the second tool), comes between them. In this configuration, if the goal position x^g is commanded directly as the MPC desired position $\bar{x}^{mpc}$, the tool would try to move directly towards the goal, stopping in a configuration with zero velocity and at a safe distance from the obstacle, that is the one depicted in Figure 3.4b.

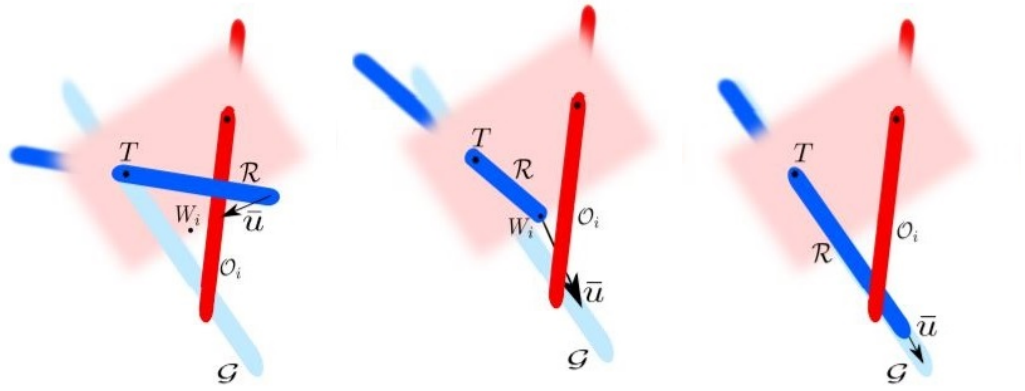


(a) The tool before the insertion movement. **(b)** The tool after the insertion movement.

Figure 3.4: Wrong tool movement which cause the obstacle to be not overtaken.

A possible solution to this problem consists in properly planning a set of waypoints towards the final target configuration to be provided as intermediate goals to the MPC. In this way, as shown in [90], it is possible to lead the robot along preferential directions

to avoid the obstacle while taking into account the RCM constraint. Starting from the same configuration described previously, reported now in Figure 3.5a, the tools can be first driven to reach the waypoint avoiding the obstacle (Figure 3.5b), and then driven to the goal position (Figure 3.5c).



(a) The tool before the movement.

(b) The tool after the movement towards the waypoint.

(c) The tool at the sesired position.

Figure 3.5: Correct tool movement which cause the obstacle to be overtaken.

The proposed planning strategy is based on simple geometric considerations and it can be executed in real-time with reduced computational overhead when compared to potential energy-based methods like [87], which is crucial for having a reactive behavior of the robotic system holding the laparoscopic tool.

The motion of each *controlled tool* is planned separately. When the motion of one of the *controlled tools* is computed, all the other tools in the workspace are considered as obstacles. All tools are wrapped into capsules and, therefore, the planning problem can be seen as the problem of planning the motion of one capsule constrained to a RCM point from an initial to a final configuration while avoiding other capsules.

In the following, $\mathcal{C}(C_1, C_2)$ denote a generic capsule, where $C_1 = (x_{C_1}(t), y_{C_1}(t), z_{C_1}(t))$ and $C_2 = (x_{C_2}(t), y_{C_2}(t), z_{C_2}(t))$ are the two end-points of the capsule that identify the pose of the capsule. The term $\overline{C_1C_2}$ denotes the axis passing through C_1 and C_2. The end-points will be omitted when clear from the context.

Let $\mathcal{R}(R_1, R_2)$, $\mathcal{G}(G_1, G_2)$ and $\mathcal{O}_i(O_{1,i}, O_{2,i})$, with $i = 0, \ldots, N_o$ be the capsules identifying the tool whose motion needs to be planned, the goal configuration of $\mathcal{R}$, and the obstacles, respectively. The planning algorithm is reported in Algorithm 1.

The necessary data are the capsules modeling the tool at the actual configuration $\mathcal{R}$, the tool at goal configuration $\mathcal{G}$, and all the obstacles $\mathcal{O}_i$, $i = 1, \ldots, N_o$.

For each obstacle, a local waypoint is generated according to the following procedure. The motion plane $\mathscr{M}$, i.e. the plane on which the tool can reach the goal configuration in case of no obstacles, is generated (Line 3). Formally, this is the plane orthogonal to the normal vector:

$$n = \frac{R_1 - R_2}{\|R_1 - R_2\|} \times \frac{G_1 - G_2}{\|G_1 - G_2\|} \tag{3.8}$$

Algorithm 1 Waypoint strategy computation

```
1:  procedure GETWAYPOINT(R, G, O_i)
2:      Q = ∅
3:      M=computeMotionPlane(R_1R_2, G_1G_2)
4:      for i ← 1, N_o do
5:          S_i = sample(O_{1,i}, O_{2,i})
6:          Z_i=getClosest(S_i, T)
7:          if isFree(O_{i,1}, O_{i,2}, R_1, M) then
8:              W_i=∅
9:          else
10:             W_i=project(Z_i, M, O_{i,1}O_{i,2})
11:         end if
12:         addWayPoint(W_i, Q)
13:     end for
14:     W=computeFinalWayPoint(Q)
15: end procedure
```

and containing $\overline{R_1R_2}$ and $\overline{G_1G_2}$.

Then a set S_i of possible escape points from the obstacle is generated by uniformly sampling the space around the endpoints of the obstacle capsule $\mathcal{O}_i$ (Line 5). Among these points, Z_i, the closest to the RCM, is chosen (Line 6) in order to give a preference to the retraction of the tool, which is often a feasible option.

If the capsule of the obstacle is parallel to the motion plane, i.e. $(O_{i,1} - O_{i,2}) \cdot n = 0$, but not contained in it, or if the capsule is not intersecting the plane, i.e. $((O_{i,1} + \sigma(O_{i,2} - O_{i,1})) - R_1) \cdot n = 0$ for all $\sigma \in [0, 1]$, then the motion plane is free and the robot can reach the desired configuration. Thus, no local waypoints are generated (Line 8).

If the obstacle intersects the motion plane, a local waypoint W_i is generated by projecting Z_i on the motion plane along the obstacle axis (Line 10), as depicted in Figure 3.6. In this way, W_i is reachable by the robot and it does not intersect the obstacle by construction.

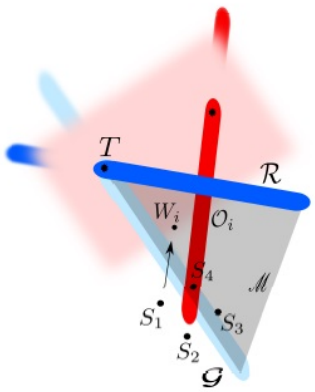


Figure 3.6: Procedure for choosing the position of the waypoint.

The set containing the local waypoints is updated (Line 12), and the global waypoint is computed as the centroid of the waypoints associated to each obstacle. Using as weight of

each waypoint β_i the inverse of the distance d_i between the tool and the related obstacle (Line 14), this can be done as:

$$W = \frac{\sum_{i=1}^{N} \beta_i W_i}{\sum_{i=1}^{N} \beta_i} \tag{3.9}$$

where N represents the cardinality of $\mathscr{Q}$ and $\beta_i = 1/d_i$.

Despite the fact that each local waypoint is external to the related obstacle, it is possible (though unlikely) that their centroid, computed as ((3.9)), could lie inside one of the obstacles: in this case, the waypoint W is set equal to the trocar, thus the MPC will compute an extraction movement, which is always collision-free.

3.7 Validation

The validation of the proposed architecture has been performed with the SARAS SOLO-SURGERY Platform. On this setup, the *controlled tools*, i.e. the tools controlled by the autonomous system, are those of the SARAS arms, while the *obstacle tools* are those of the da Vinci®.

3.7.1 Simulations

At first, a simulation environment has been developed to perform preliminary tests. In order to simulate the real movement of the robot and faithfully reproduce the real setup, a visual model and a kinematic simulator of the SARAS arms were created. The arms were simulated introducing in the simulation environment the position of the RCM and the position of the end-effector of two arms, with the relative virtual capsules.

Figure 3.7 shows the simulation environment. A SARAS arm is placed on the right side and on the left side. In red, the virtual capsules wrapping each arm's tool, and, in green, the virtual capsules wrapping the obstacles. The frames at the end of each arm represent the pose of the end-effector while the other two frames represent the goal positions.

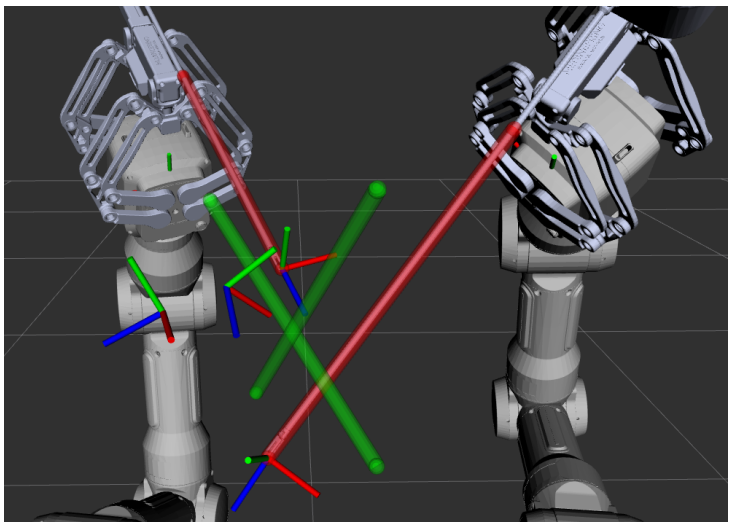


Figure 3.7: Simulation environment.

Simulations are performed providing to the system the goal configuration x_g, the initial configuration of the two arms $\bar{x}_r, \bar{x}_l$, and the initial configuration of the two obstacles. The SARAS arms and the da Vinci®capsules' radii were set to 7.5 mm, with the additional safety distance between capsules $d_s = 5\,\text{mm}$. The maximum tool velocities were set conservatively to $u_j^{\max} = 10\,\frac{\text{mm}}{\text{s}}$.

Figure 3.8 shows the results achieved using the simulation environment. The norm of the Cartesian velocity (Figure 3.8-a) of the two controlled arms clearly shows the modulation introduced by the confidence level, reported in Figure 3.8-b, underlining the capability of the controller of scaling the velocities if an uncertain situation is detected (e.g. at time $t = 13s$).

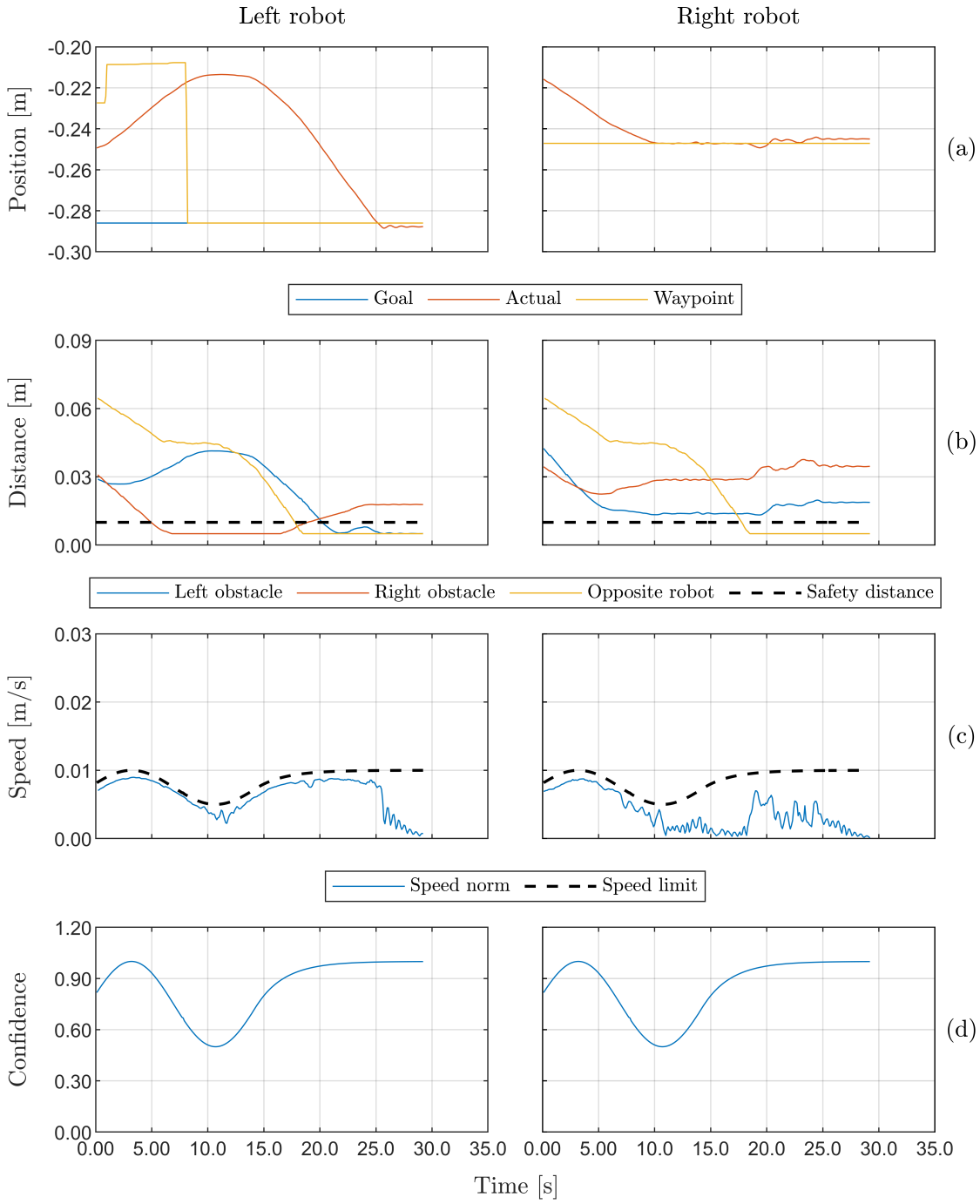


Figure 3.8: Simulation results with fixed obstacles. (a) Goal, Actual and Waypoint Cartesian positions. (b) Distances between the tools. (c) Control velocity norm. (d) Confidence.

Figure 3.8-c shows the evolution of the Cartesian positions. For the sake of clarity, only movements along one of the axis are reported (here and in all the subsequent plots). The position of the waypoint switches during the simulation in order to allow the SARAS arms to overtake the obstacles. This happens only if an obstacle needs to be overtaken, as for the left robot in the time interval $t \in [0, 25]$. If no obstacle needs to be overtaken, the waypoint is set to the target position, as for the right robot, where the waypoint and the target position overlap for the entire simulation.

Since the waypoint position is used as the reference for the MPC controller and the waypoint position converges to the target position, the overall controller allows the system to reach the target position. The real Cartesian position of the arms is not reported in the plots since the implemented simulator is purely kinematic, namely, there are no differences between the commanded position and the real position.

Thanks to the MPC controller and the waypoint motion strategy, the controller is able to perform all the movements avoiding collision between tools, as clearly visible in Figure 3.8-d where the distances between the tools are reported.

A particular behavior of the controller can be also observed in Figure 3.8. Indeed, in the first 30 seconds of the simulation, the right robot reaches the target position and starts to track it, as visible in Figure 3.8-c. At that time, the right robot moves in order to allow the left robot to reach its goal position. Indeed, the distance between the two arms goes to the minimum allowed distance, as visible in Figure 3.8-d. Finally, a new configuration is computed for both the robots in order to minimize the distance from the target position.

Simulation tests have been performed also with moving obstacles (Figure 3.9) to validate the local waypoint algorithm and the response of the MPC optimization. The obstacles pivot at a constant tool-tip linear velocity of $2\,\frac{\text{mm}}{\text{s}}$ to interfere with the initially planned waypoint. The different geometrical alignment of the tools forces the re-evaluation of a new waypoint.

A direct comparison of Figure 3.8 and Figure 3.9, subfigures (c) and (d), illustrates the adapted control strategy to the moving obstacles: as the obstacle closes to the moving tools, a different trajectory is forced.

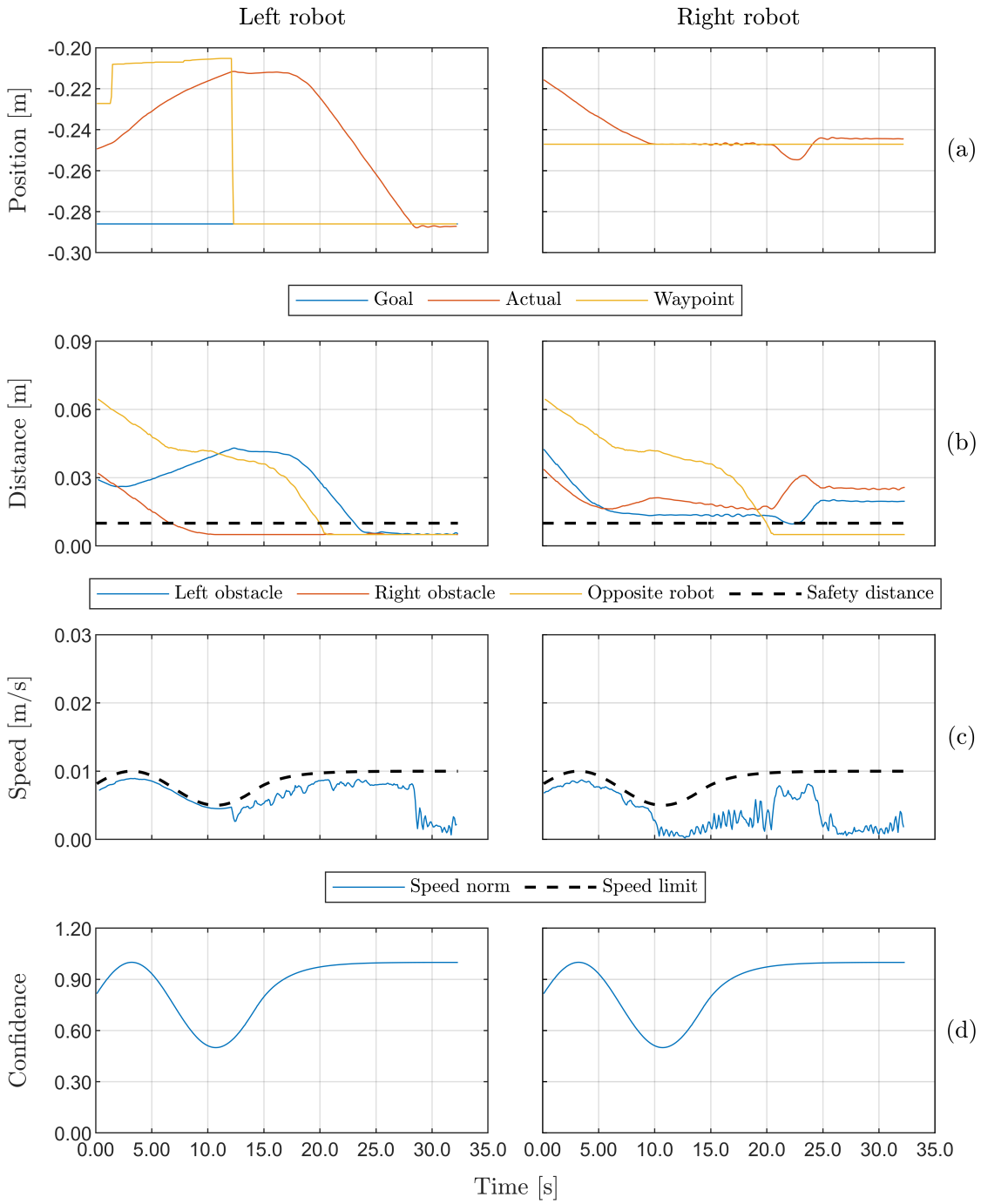


Figure 3.9: Simulation results with moving obstacles. (a) Goal, Actual and Waypoint Cartesian positions. (b) Distances between the tools. (c) Control velocity norm. (d) Confidence.

3.7.2 Experiments

As for the simulations, the experiments on the real SARAS set-up have been performed providing to the system the goal configuration x_g. The configuration of the two arms $\bar{x}_r, \bar{x}_l$ and the configuration of the two obstacles, in this case, are continuously updating using the robots readings. The two controlled arms and the two obstacles arms are initialized in such a way that each controlled arm needs to overcome an obstacle, in order to highlight the capabilities of the controller to avoid obstacles.

The radius of the capsules is set to 5.0 mm for the da Vinci®tools and to 4 mm for the SARAS tools to fit their dimensions with a little safety margin. The safety distance d_s is set to 10 mm, higher than the one used in the simulated environment to take into account possible calibration inaccuracies.

Figure 3.10 shows the results achieved using the real setup and confirms the results obtained in simulation. Figure 3.10-a reports the Cartesian velocities of the two controlled arms while Figure 3.10-c reports their Cartesian positions. It is worth highlighting that the noise in the velocities in Figure 3.10-a is due both to the numerical derivation of positions measured by potentiometers (and not encoders) and by the fact that the RCMs are virtually (via software) but not physically present. With real trocars, the shaking of such slender (and not collocated) tools would be drastically reduced.

Good tracking performances can be appreciated looking at the small difference between the commanded Cartesian position and the real Cartesian position (red lines and orange lines in Figure 3.10-c). This shows that the robots implementing the MPC commands reach their target positions while avoiding the obstacles, thanks to the waypoint motion strategy described in Section 3.6.

All the movements are performed avoiding collisions, as clearly appreciable in Figure 3.10-d, and modulating the velocities with respect to the confidence level provided to the Control module.

The same position was commanded as goal configuration for the two *controlled tools*. Referring to Figure 3.10, the position of the waypoints is computed as an intermediate point for both the controlled arms. The *controlled tools* move to the waypoints. Then, the waypoints switch to the target positions, driving the robots towards the goal configurations since obstacles are assumed to have been already overcome. Since the goal configuration is the same for both the tools, neither of them reach the target position since a collision would occur. The system converges to a configuration where the distance between the actual position and the desired one is minimized but collisions are avoided, as observed also in simulation.

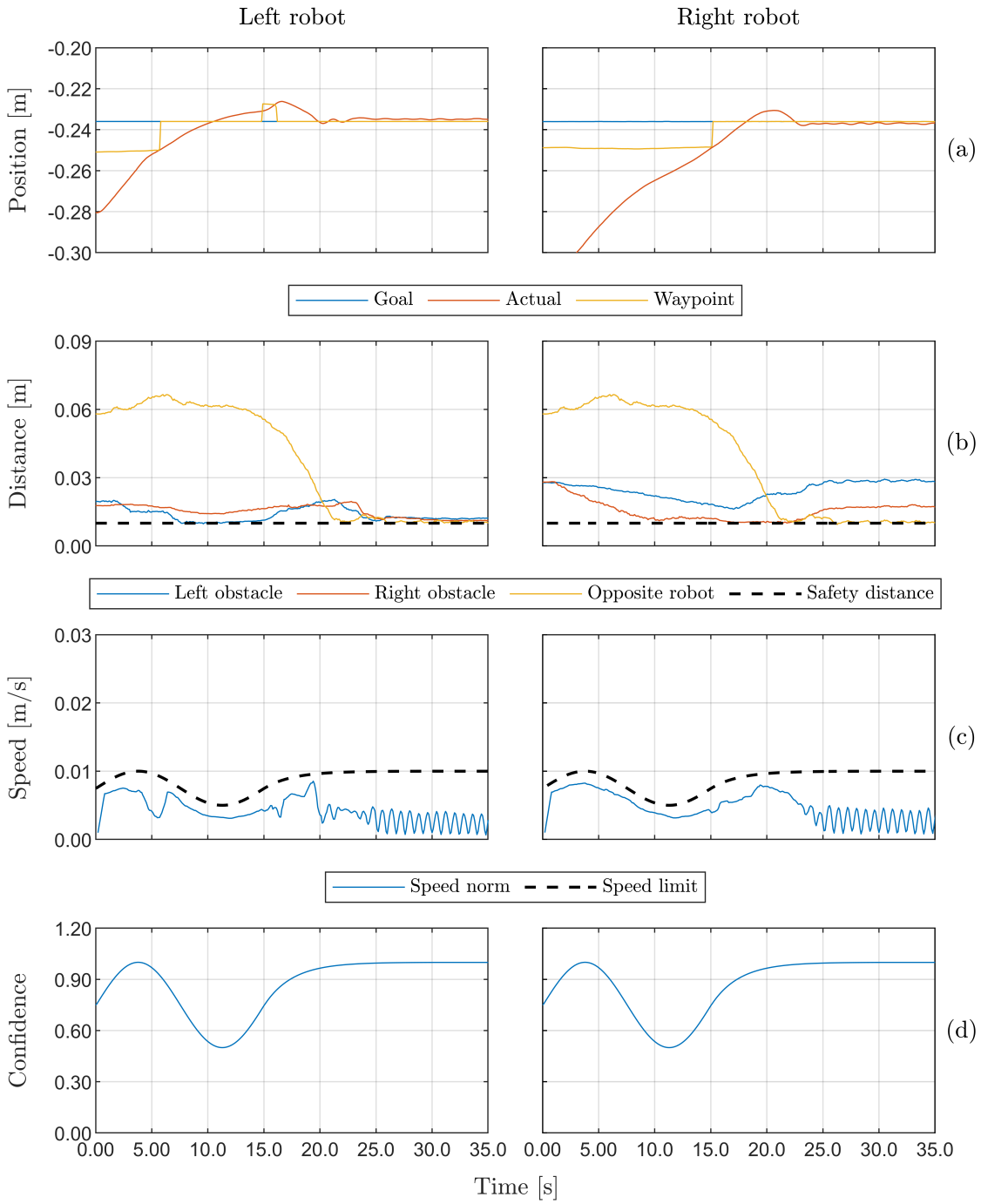


Figure 3.10: Experiment results. (a) Goal, Actual and Waypoint Cartesian positions. (b) Distances between the tools. (c) Control velocity norm. (d) Confidence.

3.8 Model Predictive Controller Improvements

The nature with which the constraints were modeled so far led to the design of a non-linear MPC. It is well known that non-linearities are difficult to treat, especially in the case of optimization-based systems. Moreover, the complexity introduced by the prediction of the model over time led to the following problems:

- the optimization process was computationally demanding so that the controller was able to execute online at a maximum frequency of only 10 Hz, unsuitable for reaching the required reactivity.
- the quality of the optimization process was low as it was necessary to decrease the number of iterations in the optimization process to make the controller usable online. This led to non-smooth trajectories and jerky movements.

This section reports all the improvements to address the aforementioned problems. In particular, the work focuses on the development of a linear form of the speed and collision constraints. In fact, a linear form of the constraints allows a linear formulation of the MPC problem, which is much easier and faster to solve. Moreover, the confidence level $\alpha(k)$ is omitted in the following since it does not affect the performance and can be easily integrated.

Since the model (3.2) is already linear, the model embedded MPC does not need to be modified.

3.8.1 Constraints

Velocity Limit Constraint

Equation (3.4), which describes the velocity limit constraint implemented so far, is clearly non-linear since the norm is a non-linear function. This means that this type of constraint cannot be used in a linear MPC.

However, it can be noticed that Equation (3.4) indicates that $\bar{u}_j$ lies within a sphere $\mathscr{S}$ of radius u_j^{max} centered in $\bar{0} \in \mathbb{R}^3$. Therefore, a linear description of the encompassed spherical region also summarizes the constraint in (3.4): this can be achieved by following Algorithm 2.

Algorithm 2 Velocity limit constraint linear approximation

procedure GETSPEEDCONSTRAINT(N_p)
  $\mathscr{P} \leftarrow sample(\mathscr{S}, N_p)$
  $\mathscr{T} \leftarrow triangulate(\mathscr{S}, \mathscr{P})$
  for $i \leftarrow 1, size(\mathscr{T})$ **do**
    $plns(i) \leftarrow computePlanes(t)$
  end for
  $(\bar{A}_j^u, \bar{b}_j^u) \leftarrow groupPlanes(plns)$
  return $\bar{A}_j^u, \bar{b}_j^u$
end procedure

Start by sampling the sphere $\mathscr{S}$ with $N_p \in \mathbb{N}$ point, obtaining the set of points $\mathscr{P}$ lying on $\mathscr{S}$ (Line 2).

Then, compute the triangulation $\mathscr{T}$ of the convex hull of $\mathscr{P}$ (Line 3). Each triangle i in the triangulation $\mathscr{T}$ identifies a plane with equation:

$$\lambda_j^i \, {}^x\bar{u}_j + \beta_j^i \, {}^y\bar{u}_j + \zeta_j^i \, {}^z\bar{u}_j + \rho_j^i = 0 \tag{3.10}$$

where λ_j^i, β_j^i, ζ_j^i, and $\rho_j^i \in \mathbb{R}$ are the plane coefficients and ${}^x\bar{u}_j$, ${}^y\bar{u}_j$ and ${}^z\bar{u}_j \in \mathbb{R}$ are the Cartesian components of $\bar{u}_j$.

Grouping up all the planes identified by each triangle in the triangulation $\mathscr{T}$ (Line 7) leads to:

$$\bar{A}_j^u \bar{u}_j = \bar{b}_j^u \tag{3.11}$$

where:

$$\bar{A}_j^u = \begin{bmatrix} \lambda_j^1 & \beta_j^1 & \zeta_j^1 \\ \vdots & \vdots & \vdots \\ \lambda_j^{N_p} & \beta_j^{N_p} & \zeta_j^{N_p} \end{bmatrix} \quad \bar{b}_j^u = \begin{bmatrix} -\rho_j^1 \\ \vdots \\ -\rho_j^{N_p} \end{bmatrix} \tag{3.12}$$

It is worth noting that the set of planes described in Equation (3.11) can be used to describe a polygonal approximation of the velocity constraint by setting:

$$\bar{A}_j^u \bar{u}_j \leq \bar{b}_j^u \tag{3.13}$$

It is necessary to underline that the approximation with which the constraint is described increases in accuracy the higher the value of N_p, i.e. the number of points with which the sphere $\mathscr{S}$ is sampled. The value of N_p also coincides with the number of equations with which the constraint is approximated, as in Equation (3.13). This constraint will also be projected along the prediction horizon of the MPC, meaning that too many points cannot be used with the aim of preserving performances.

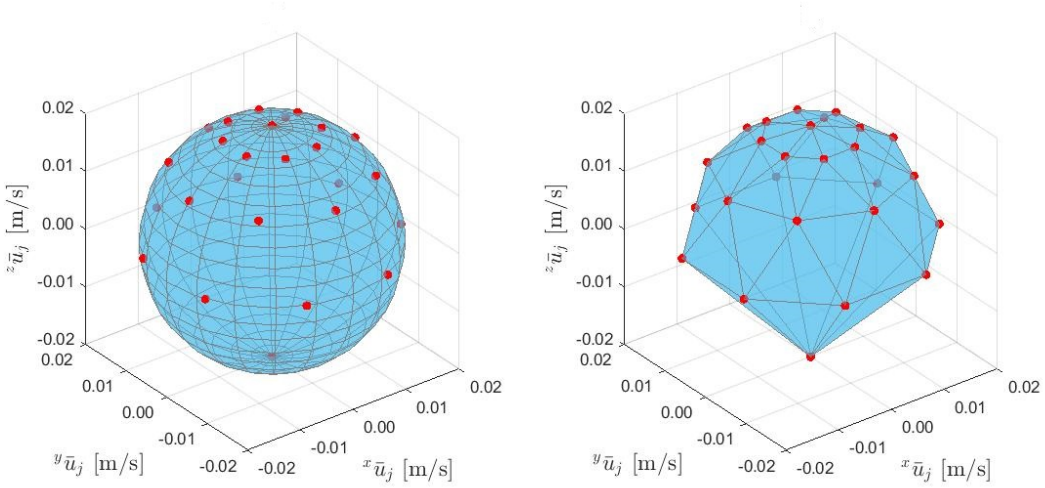
A uniform sampling with a small number of points can lead to a computationally feasible set of constraints but to a rough approximation of the velocity constraint. In order to have a more accurate approximation around the direction of motion, a non-homogeneous sampling of the sphere can be adopted. This will lead to a higher number of points, i.e. a better approximation of the velocity constraint, around the direction the tool is moving and, at the same time, a limited global number of points, i.e. a limited computational cost. This procedure is reported in Algorithm 3.

Start by choosing a small number of points N_p and by selecting a sample velocity, being equal to $\bar{u}_j^\star = [0, 0, \bar{u}_j^{\max}]$, i.e. the input velocity aligned to the positive z-axis with the maximum value (Line 2).

Distribute the set of points $\mathscr{P}$ in order to be more concentrated in the region of the sphere $\mathscr{S}$ closer to $\bar{u}_j^\star$, while paying particular attention to position a point in $-\bar{u}_j^\star$ (Line 3). Indeed, the introduction of this point allows the system to reverse its speed whenever an obstacle is encountered.

Algorithm 3 Velocity limit constraint simplified

1: **procedure** GETSPEEDCONSTRAINTS(N_p, $\bar{u}_j(t-\Delta t_c)$)
2: $\quad \bar{u}_j^\star \leftarrow [0, 0, \bar{u}_j^{\max}]$
3: $\quad \mathscr{P} \leftarrow sampleMod(\mathscr{S}, N_p, \bar{u}_j^\star)$
4: $\quad \mathscr{T} \leftarrow triangulate(\mathscr{S}, \mathscr{P})$
5: $\quad \mathscr{T} \leftarrow rotate(\mathscr{T}, \bar{u}_j(t-\Delta t_c))$
6: $\quad$ **for** $i \leftarrow 1, size(\mathscr{T})$ **do**
7: $\quad\quad plns(i) \leftarrow computePlanes(t)$
8: $\quad$ **end for**
9: $\quad (\bar{A}_j^u, \bar{b}_j^u) \leftarrow groupPlane(plns)$
10: $\quad$ **return** $\bar{A}_j^u, \bar{b}_j^u$
11: **end procedure**



(a) Original constraint, coincident with a sphere.

(b) Approximated constraint, coincident with a polygonal shape.

Figure 3.11: Illustration of the velocity limit constraint.

Next, compute the triangulation $\mathscr{T}$ 4 and apply a rigid rotation to the triangulation to align the sample velocity $\bar{u}_j^\star$ to the most likely one, i.e. the solution at the second step of the prediction horizon of the optimal solution at time $t-\Delta t_c$ (Line 4). This generates the constraint (3.13).

Finally, end the procedure as the previous one by grouping up all planes (Line 9).

Figure 3.11 shows an illustration of the procedure described above, which can be simply implemented to both the *controlled tools*. The blue shape represents the constraint while the set of points used for the linear approximation is reported in red.

Collision Avoidance Constraint

In order to set the minimum distance between the instruments, Equation (3.5) exploits the distances between capsules. As for the constraint on the maximum speed, this constraint is also non-linear since the distance between capsules is a non-linear function of the position of the instruments. Therefore, it is necessary to find a linear formulation that approximates this constraint.

Consider now the problem of moving the j-th *controlled tool* in such a way collisions with the w-th *obstacle tool* are avoided. Build a capsule model around each of them to identify the following remarkable quantities:

- p_j^w: the segment representing the capsule distance between the j-th *controlled tool* and the w-th *obstacle tool*.
- $Pa_j^w \in \mathbb{R}^3$: the point on the axis of the w-th *obstacle tool* capsule which intersects p_j^w.
- $Pc_j^w \in \mathbb{R}^3$: the point on the capsule of the w-th *obstacle tool* capsule which intersects p_j^w.
- $\hat{p}_j^w \in \mathbb{R}^3$: the unit vector lying on p_j^w and pointing from the w-th *obstacle tool* to the j-th *controlled tool*, which is:

$$\hat{p}_j^w = \frac{Pc_j^w - Pa_j^w}{||Pc_j^w - Pa_j^w||} \tag{3.14}$$

Consider now the plane π with normal coincident with $\hat{p}_j^w$ and passing through a point $P^\star \in \mathbb{R}^3$ defined as:

$$P^\star = Pc_j^w + d_{sft}\,\hat{p}_j^w \tag{3.15}$$

where $d_{sft} \in \mathbb{R}+$ is a positive definite safety distance which takes into account the size of the capsules and the distance to keep between the tools.

The plane π will have an equation like:

$$\alpha_j^w\,{}^x\bar{x}_j + \gamma_j^w\,{}^y\bar{x}_j + \eta_j^w\,{}^z\bar{x}_j + \phi_j^w = 0 \tag{3.16}$$

where α_j^w, γ_j^w, η_j^w, and $\phi_j^w \in \mathbb{R}$ are the plane coefficients and ${}^x\bar{x}_j$, ${}^y\bar{x}_j$, and ${}^z\bar{x}_j \in \mathbb{R}$ are the x, y, and z Cartesian coordinates of the position of the j-th *controlled tool*.

Since the laparoscopic tools have one of the extremes fixed, which is the RCM, and it is basically a straight rod, one way to set that the *controlled tool* will always remain at a minimum safe distance from the *obstacle tool* is to impose that the end-effector of the *controlled tool* lies in the region of space identified by the plane π in which there is not the *obstacle tool*. This can be done by setting:

$$\alpha_j^w\,{}^x\bar{x}_j + \gamma_j^w\,{}^y\bar{x}_j + \eta_j^w\,{}^z\bar{x}_j + \phi_j^w \leq 0 \tag{3.17}$$

Let $N_o \in \mathbb{N}$ be the number of *obstacle tools* in the workspace, the collision avoidance constraint for the j-th *controlled tool* with all the *obstacle tools* can be obtained by extending what said before to every obstacle. Hence, by defining:

$$\bar{A}_j^x = \begin{bmatrix} \alpha_j^1 & \gamma_j^1 & \eta_j^1 \\ \vdots & \vdots & \vdots \\ \alpha_j^{N_o} & \gamma_j^{N_0} & \eta_j^{N_0} \end{bmatrix} \quad \bar{b}_j^x = \begin{bmatrix} -\phi_j^1 \\ \vdots \\ -\phi_j^{N_o} \end{bmatrix} \tag{3.18}$$

Finally, the collision avoidance constraint can be formulated as a linear inequality as:

$$\bar{A}_j^x \bar{x}_j \leq \bar{b}_j^x \tag{3.19}$$

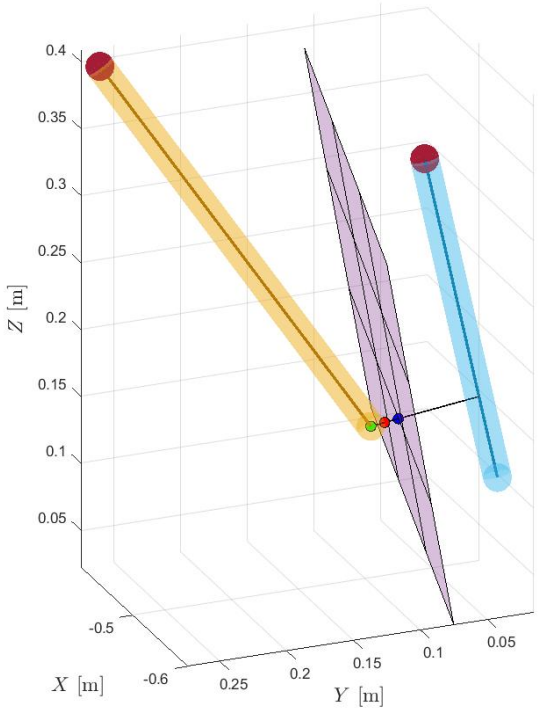


Figure 3.12: Illustration of the collision avoidance constraint.

As for the velocity limit constraint presented in the previous subsection, this procedure can be simply extended to both the *controlled tools* too.

It is worth noting that among the set of tools that each *controlled tool* considers as an obstacle, the other *controlled tool* must also be considered, e.g. the right *controlled tool* must consider the left *controlled tool* as an obstacle.

Figure 3.12 reports an illustration of the plane used to describe the collision avoidance constraint for a *controlled tool* and an *obstacle tool*. The blue and the yellow capsules are built around the *controlled tool* and the *obstacle tool* respectively, with the RCMs highlighted in dark red. Each capsule reports also its axis. The points Pa_j^w, Pc_j^w, and $P^\star$ are reported in green, red, and blue, respectively.

3.8.2 Cost Function

The cost function of the MPC as in (3.6) is a quadratic formulation that can be easily implemented in a linear QP problem. With respect to (3.6), a slack variable is added for softening some of the constraints. The softening of the constraints is a well-known technique to handle constraints (see for example [91]) and, in this case, is necessary to allow the correct operation of the linear solver.

The new cost function is defined as:

$$J(x^{\mathrm{MPC}}, x) = \sum_{i=0}^{p-1} ||x^{\mathrm{MPC}}(k) - \hat{x}(k+i)||^2 + \sum_{i=0}^{p-1} \epsilon(k+i)^2 \tag{3.20}$$

where $\epsilon(k+i) \in \mathbb{R}^{2N_o+2}$ is the slack variable for soft constraining at the time step $k+i$.

The size of the slack variable $\epsilon(k+i)$ is due to the fact that the constraint softening will be applied only to collision avoidance constraints, as will be clearer in the next section. This choice is given by the nature of the collision avoidance constraint, which reports the safety distance parameter d_{sft}. This means that a small variation of the constraint introduced by the slack variable ϵ can be easily compensated by increasing d_{sft} without compromising the performances. This does not hold for the speed limit constraint.

3.8.3 MPC formulation

The following linear constrained finite-horizon optimal control problem can be now set:

$$\begin{aligned} \min_{\mathbf{u}} \ & \sum_{i=0}^{p-1} ||x^{\mathrm{MPC}}(k) - \hat{x}(k+i)||^2 + \sum_{i=0}^{p-1} ||\epsilon(k+i)||^2 \\ s.t. \ & \hat{x}(k+i+1) = \hat{x}(k+i) + Bu(k+i) \\ & A^u(k+i)\hat{u}(k+i) < b^u(k+i) \\ & A^x(k+i)\hat{x}(k+i) - \epsilon(k+i) < b^x(k+i) \\ & i = 0, ..., p-1 \end{aligned} \tag{3.21}$$

where:

$$A^u(k+i) = \begin{bmatrix} \bar{A}^u_r(k+i) & 0 \\ 0 & \bar{A}^u_l(k+i) \end{bmatrix} \quad b^u = \begin{bmatrix} \bar{b}^u_r(k+i) \\ \bar{b}^u_l(k+i) \end{bmatrix} \tag{3.22}$$

and:

$$A^x(k+i) = \begin{bmatrix} \bar{A}^x_r(k+i) & 0 \\ 0 & \bar{A}^x_l(k+i) \end{bmatrix} \quad b^x = \begin{bmatrix} \bar{b}^x_r(k+i) \\ \bar{b}^x_l(k+i) \end{bmatrix} \tag{3.23}$$

As for the non-linear MPC provided in Section 3.5, the solution of (3.21) provides the optimal control input sequence $\mathbf{u} = [u(k), ..., u(k+p-1)]$. The first component $u(k)$ of $\mathbf{u}$ is then used to compute the desired motion position $x^d(k+1) \triangleq x(k) + Bu(k)$ to be commanded to the robots.

The speed limit constraint (3.22) is projected along the prediction horizon by considering the same velocity limit constraint approximation on the entire prediction horizon:

$$A^u(k+i) = A^u(k) \quad \forall i = 0, ..., p-1 \tag{3.24}$$

The collision avoidance constraint (3.23) is projected along the prediction horizon by calculating the position of the obstacles as if they were moving at a constant speed and using the position obtained in the previous optimization step as the position for the *controlled tools*.

Both choices were made to preserve the linearity of all constraints in the optimization problem.

3.9 Validation

The validation of the proposed architecture has been performed with the SARAS LAPARO2.0-SURGERY Platform. On this setup, the *controlled tools*, i.e. the tools controlled by the autonomous system, are those of the SARAS arms, while the *obstacle tools* are two manual tools.

3.9.1 Simulations

Before moving on the real set-up, several simulations were performed to test the effectiveness of the proposed architecture, and to compare the performance achieved with the new controller with respect to the previous one. The main parameters observed for the comparison were the average optimization time per cycle, which evaluates the real-time applicability, and the quality of the solution in terms of smoothness.

Both parameters were evaluated on a total of 60 simulations, 30 of those with the new controller (which will be referred to from now on as *linear MPC*), and 30 with the previous controller (which will be referred to from now on as *non-linear MPC*). For both the MPCs, the simulations were divided into blocks of 10, each with a different prediction horizon p, for a total of 3 prediction horizons.

The 10 simulations of each block were performed providing 10 different starting positions for both the *controlled tools* and the same goal. All positions have been chosen in such a way that they are all realistically plausible with the real set-up. The movement of the *obstacle tools* was acquired once through the real set-up and faithfully reproduced in all simulations. Finally, to ensure that the prediction horizon was the same regardless of the optimization time, all the simulations were performed with the same cycle time, set equal to $\Delta t_c = 10\,\text{ms}$.

Figure 3.13 reports the mean and standard deviation of the optimization times recorded for each prediction horizon, for the *linear mpc* and for the *non-linear mpc*.

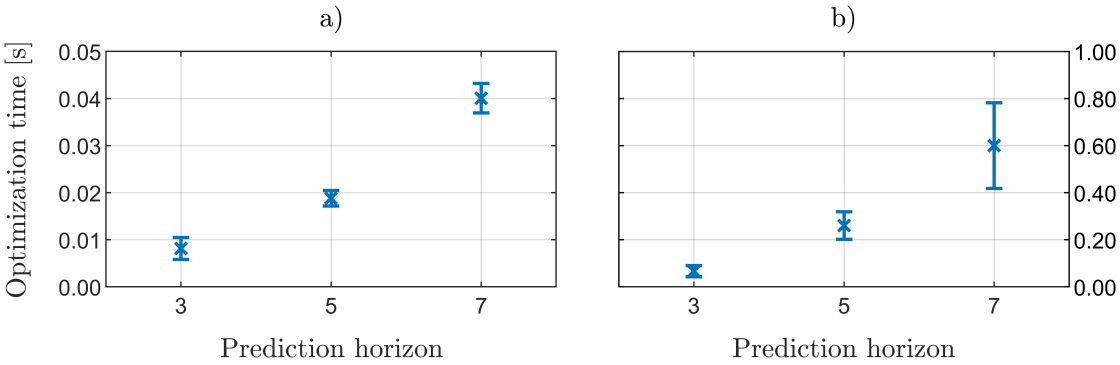


Figure 3.13: Comparison of the optimization times. a) *linear mpc*. b) *non-linear mpc*.

As visible from Figure 3.13, the optimization times obtained with the *linear mpc* are drastically lower than those obtained with the *non-linear mpc*. Furthermore, observing the degree with which the slope of the optimization time rises, the *linear MPC* results more

scalable than the *non-linear MPC*, reporting a lower slope and lower cycle times for each prediction horizon. Both these results state a significant improvement in performance with respect to the previous controller.

Figure 3.14 shows the results obtained in terms of smoothness of the trajectory, reporting the mean and the standard deviation of Spectral Arc Length (SAL) [92] of the trajectory, for each prediction horizon, and for the *linear mpc* and the *non-linear mpc*. The SAL is chosen as the smoothness measure since it is proven to be an effective metric for the smoothness of a trajectory, with respect to, for example, the normalized mean absolute jerk, the number of peaks in the speed profile or the normalized mean speed. It is also more robust to noise.

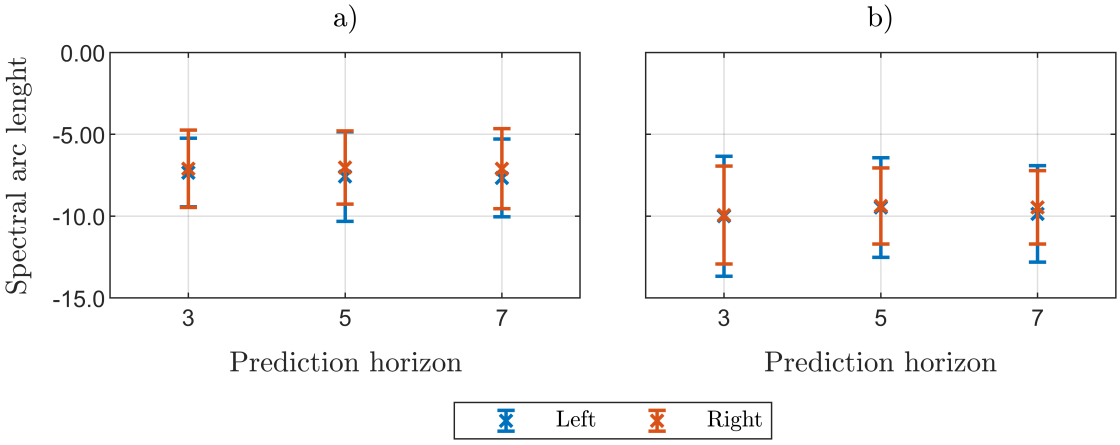


Figure 3.14: Comparison of the trajectory smoothness. a) *linear mpc*. b) *non-linear mpc*.

As shown in Figure 3.14, there are no appreciable differences in the SAL changing the prediction horizon and the type of MPC used. This means that the quality of the trajectory has not been degraded using the *linear mpc*, which uses the linear approximation of the constraints. This further states the effectiveness of the proposed system since the trajectory returned by the *non-linear MPC* provides the best ground truth for the smoothness of the trajectory, considering the constraint as they really work.

It is important to underline that these results are not in contrast with what is reported in Section 3.8, i.e. the non-linear MPC provides non-smooth trajectory. Indeed, the problems related to the quality of the trajectory is related to the practical implementation, where the cycle time must be set, at least, to the optimization time.

3.9.2 Experiments

This section reports the results related to the practical implementation of the two types of MPCs. Three different experiments are reported:

1. The *non-linear MPC* with prediction horizon $p = 3$ and cycle time $\Delta t_c = 0.066$ ms (15 Hz). This experiment implements the *non-linear MPC* with the best settings.
2. The *linear MPC* with prediction horizon $p = 3$ and cycle time $\Delta t_c = 0.008$ ms (120 Hz). This experiment implements the *linear MPC* with the same settings as the previous experiment.

3. The *linear MPC* with prediction horizon $p = 7$ and cycle time $\Delta t_c = 0.04\,\text{ms}$ (25 Hz). This experiment implements the *linear MPC* with the best settings.

All the experiments are performed with different starting positions for both the *controlled tools* and the *obstacle tools*. The *obstacle tools* are moved during the experiment in such a way to hinder the movement of the *controlled tools* towards the goal. Once the goal was reached the *obstacle tool* is used to disturb the equilibrium position and test the reactivity of the system. All the cycle times are set according to the optimization times obtained in the simulations and shown in Figure 3.13.

Figure 3.15 reports the results of the experiments in terms of smoothness. Comparing the results of experiments 1 and 2, which refer to the two types of MPCs with the same settings, a significant increase in the smoothness of the trajectory when passing from the *non-linear MPC* to the *linear MPC* can be noted. This improvement is mainly related to the shorter cycle time with which the *linear MPC* is able to solve the optimization problem and control the robots.

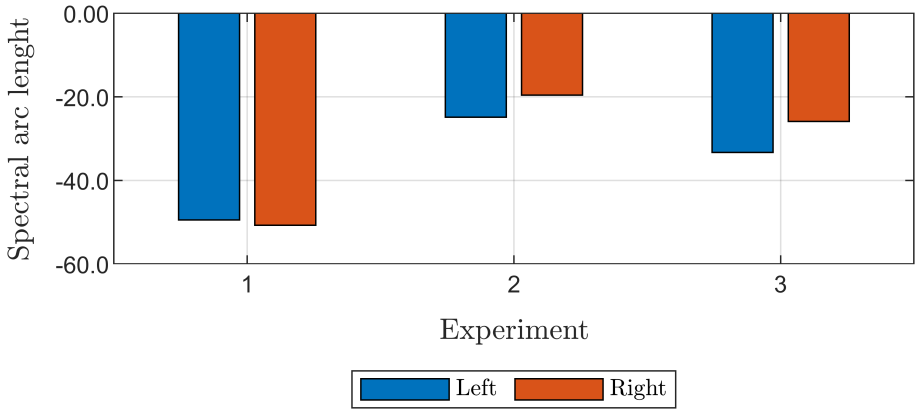


Figure 3.15: Comparison of the trajectory smoothness.

The *linear MPC* with the best settings (i.e. experiment 3) reports a lower value of smoothness than in experiment 2 but evidently higher with respect to experiment 1. Once again, the increased cycle time decreases the overall quality of the trajectory. In fact, the cycle time in experiment 1 and experiment 3 become comparable. However, the greater number of samples in the prediction horizon p with the *linear MPC* of experiment 3 allows reaching a higher level of smoothness with respect to the *non-linear MPC*, underlining the effectiveness of the proposed method.

Furthermore, this result allows to state that, as already observed in the simulations, the linear approximation of the constraints does not introduce degradation in terms of quality of the trajectory.

Figure 3.16 reports the results of experiment 3 in terms of motion. All the following considerations hold for all the experiments. For clarity of presentation, the movement of only one axis of each *controlled tool* is reported.

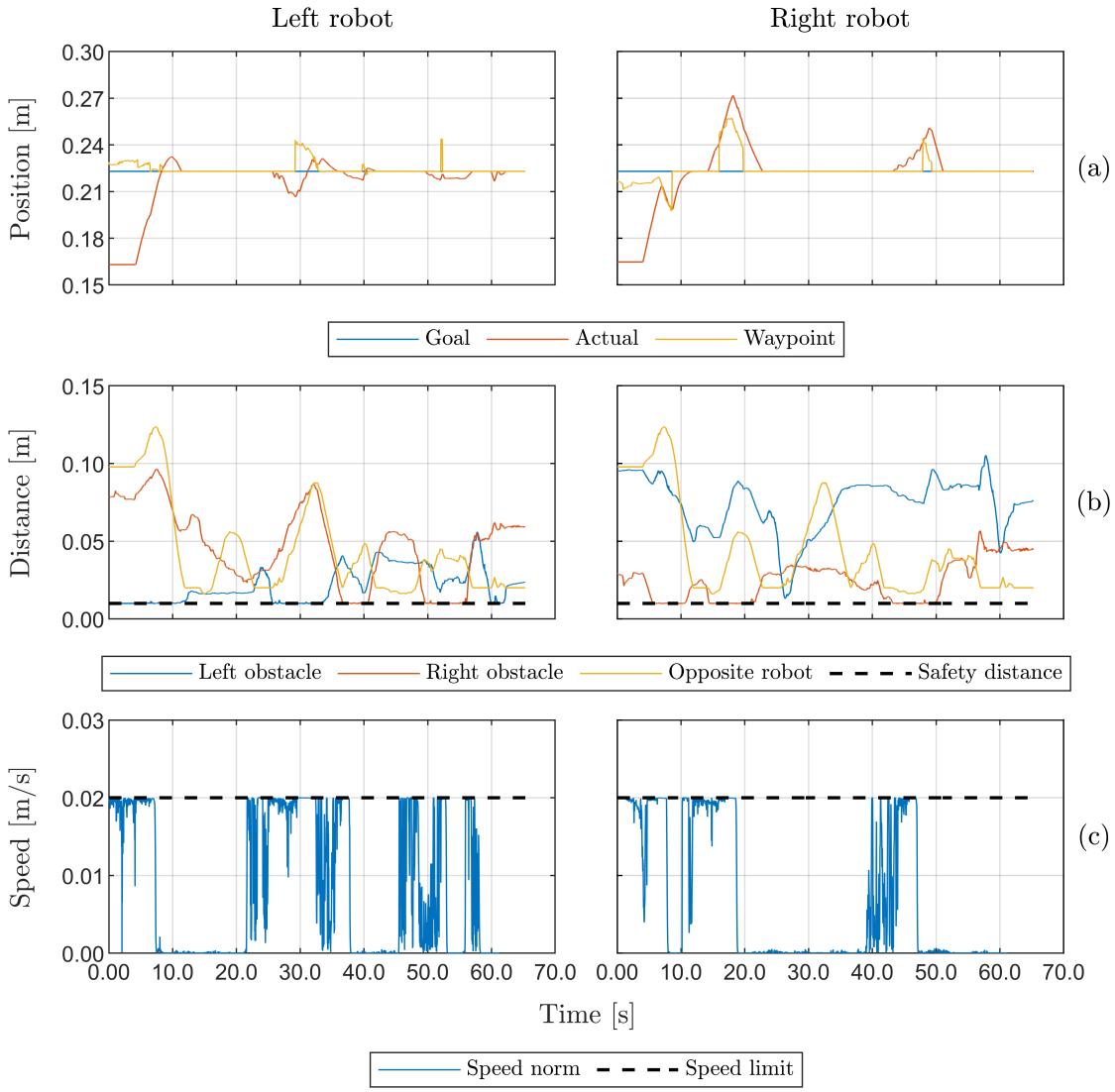


Figure 3.16: Experiment results. (a) Goal, Actual and Waypoint Cartesian positions. (b) Distances between the tools. (c) Control velocity norm.

Figure 3.16 shows that the system works correctly. Both the *controlled tools* start from their initial positions and move to the waypoint. This behavior is particularly visible at $6\,\text{s} \leq t \leq 9\,\text{s}$ and $43\,\text{s} \leq t \leq 57\,\text{s}$ for the right *controlled tool*, where the actual position coincides with the waypoint position for about 3 s and 4 s. The waypoint tends to coincide with the goal when the geometric conditions allow the *controlled tools* to reach the goal without collisions, bringing the system to the required position.

Figure 3.16 shows also that the movements of the tools are always collision-free: the safety distance $d_{sft} = 0.01\,\text{m}$ computed as the distance between capsules is always guaranteed.

The commanded speed also respects the conditions of correct system operation. in fact, the current speed norm never exceeds the limit $\bar{u}_j^{max} = 0.02\,\text{m/s}$.

3.10 Conclusions

This chapter deals with the development of the Planning and Navigation module of the SARAS cognitive layer, which allows the real-time coordination of multiple laparoscopic tools toward their target positions guaranteeing collision-free trajectory. Also in this case, the discussion of the proposed architecture is general, i.e. independent of the number of laparoscopic tools involved in the surgical procedure, and has been optimized to improve the overall performance and to make it work in real-time.

The first validation confirmed the effectiveness of the proposed architecture but also showed high computation times and incompatibility for real-time implementation. These problems were addressed by proposing a linear formulation of the constraints, which allowed the implementation of a linear MPC. The ease with which the linear problem can be solved with standard optimizers has made it possible to achieve compatibility for real-time implementation.

Moreover, The results show that there is no loss of quality of the solution when moving from the non-linear formulation, which considers the constraints as they really act, to the linear one, which exploits an approximation of the same constraints.

Chapter 4

PCNL

This Chapter 4 reports the development of a robotic assistance architecture for the renal access procedure during PCNL. In the pre-operative phase, the model of the anatomical parts involved in the procedure are segmented and converted to a 3D model, with which the insertion trajectory is planned. In the intra-operative phase, the surgeon visualizes the 3D model and the planned trajectory by means of an AR headset and performs the renal access with the assistance of the robot, on which the surgical tool is mounted. The system is validated through experiments performed by a sample of multiple users.

The work presented in this chapter is published in [5, 6, 7].

4.1 Introduction

According to the European Urology Association and the American Urological Association, PCNL is considered the gold standard for the treatment of patients with renal stones larger than 20 mm in diameter[93]. Its popularity and acceptance among urologists and patients is largely due to the fact that it is minimally invasive and is associated with low morbidity [94].

The procedure consists of inserting a specific tool (the nephroscope) into the kidney through a percutaneous track created in the patient's lower flank or abdomen. PCNL was first performed in Sweden in 1973 as a less invasive alternative to open surgery on the kidneys. Using fluoroscopy and ultrasonic imagery the urologist inserts a hollow needle into one of the renal calyxes of the kidney. This passage is then dilated, and a percutaneous sheath is inserted to accommodate the nephroscope.

The success and treatment outcomes of the surgery are very well known to be highly dependent on the precision and accuracy of the puncturing step since it must allow reaching the stone with a precise and direct path [95]. Thus, performing the renal access during PCNL is the most crucial and challenging step of the procedure with the steepest learning curve. Indeed, it has been shown that a fellow in endourology, with no previous experience in performing solo PCNL, requires 45 and 105 operations to achieve competence and excellence, respectively, in performing the procedure [96].

One reason why it is challenging for urologists to achieve good proficiency is the fact that not all training programs during the residency necessarily have faculty who perform this procedure. Indeed, in 2011 fewer than half (47%) of graduating U.S. chief residents indicated that PCNL access was routinely obtained by urologists at their institution and this limits the opportunity to pick up the required skills in practice. A 2014 review of case logs from certifying and recertifying urologists found that only 6% performed more than 10 PCNLs during the prior six months and urologist-obtained access only occurred in 20% of these cases, while in the remaining cases the renal access was performed by the radiologist [97].

Urologists have introduced various simulation models to help trainees achieve competency level in a shorter length of time. Simulators used for the assessment of PCNL skills include human cadavers, animal tissues [98], and Virtual Reality (VR) simulators to simulate human patients [99][100][101]. A recent review of the current training method has been presented in [102], before introducing The Marion K181 PCNL VR simulator. Virtual reality-based simulators have been shown to improve the performance of the trainee but they often lack realistic haptic feedback, which is key in improving the skills of the trainee [103].

On the other side, the availability of preoperative information (CT or MRI) of the anatomical parts has been exploited for implementing AR solutions for augmenting the capabilities of the surgeon during the procedure. In particular, in [104] and [105] AR systems are used to support the percutaneous access in percutaneous nephrolithotomy by overlaying a 3D model onto the image from a tablet camera.

In [106] the challenge of obtaining an appropriate access to the renal stones in the case of complex anatomical or pathological conditions is addressed. A 3D model generated from MRI is used for trajectory planning and intra-operative augmentation of real-time intraoperative ultrasonography. A recent review on augmented reality systems in surgery is presented in [107]. For experienced surgeons, augmented reality visualization of renal anatomy has been proved to be sufficient to estimate correct needle angles to reach the target. However, several doubts persist whether urologists in training still need additional technical assistance apart from AR support [104].

A different solution for precise percutaneous access in PCNL is represented by robotic-assisted systems. PAKY (Percutaneous Access to the Kidney) is a robotic system developed for PCNL to accurately position and insert a needle percutaneously into the kidney [108]. Lately, several works have been presented concerning the development of US-guided robotic systems [109]. The common approach consists of a surgical needle attached to a robotic arm that is driven, automatically or teleoperated, by the surgeon, in a 3D or 2D imaging volume [95].

More recently, in [106], a collaborative robotic-assisted system is proposed to help the surgeon correctly locate the ultrasound probe during the intervention, while the respiratory motion of the patient is compensated. Once the probe is located, the needle insertion is performed. However, a completely autonomous, semi-autonomous or teleoperated robotic system would not help residents or novel surgeons during their training, since the manual surgical intervention would be quite different from the one performed with the robot assistance.

In this work, the potentialities of AR support are blend with the technical assistance that can be provided through robotic systems, proposing an innovative solution that can assist: (i) an expert surgeon in improving the performance of the surgical intervention, by augmenting his/her capabilities (ii) a novel surgeon in strongly reducing his/her learning curve, since the system can provide assistance during the most critical phases of the intervention, especially at the early end of the learning curve when mistakes are certainly more likely.

The proposed system will allow to plan the surgical intervention in a pre-operative phase and to assist the surgeon in the intra-operative phase. In particular, the surgeon, wearing an AR headset, will directly see on the body of the patient a reconstructed 3D model of the anatomical parts of interest and the planned trajectory to be followed. Then, he/she will manually guide a robot that holds the nephroscope and will be assisted by feeling virtual forces (*virtual fixtures* [58]) that will keep him/her on the right renal access. The proposed system is validated on a robotic setup and with 11 users performing a task that emulates the renal access in PCNL.

4.2 System Architecture

The proposed system architecture, summarized in Figure 4.1, consists of an AR application to provide the surgeon with interactive 3D visualizations in all the phases of the operation, and a robotic system to assist and guide the surgeon towards the optimal execution of the intervention. In particular, for each surgical operation, two different phases can be distinguished: a *pre-operative phase*, where the 3D model that will be used for the visualization is built and the pre-operative planning of the desired trajectory is performed, and an *intra-operative phase*, where the surgeon performs the surgical intervention by wearing the AR device and with the assistance of the robotic system.

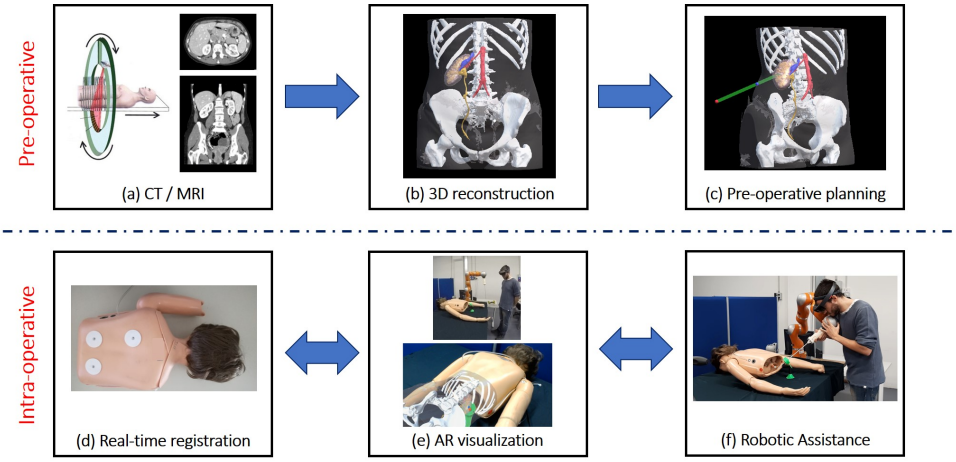


Figure 4.1: Conceptual scheme of the proposed architecture.

4.2.1 Pre-Operative Phase

The planning of a surgical intervention is typically done on the basis of pre-operative images acquired as CT or MRI of the interesting area. Starting from the CT of the area (Figure 4.1(a)), the 3D model of the interested area and the organs that are involved in the procedure is reconstructed, both as target or forbidden regions, namely anatomical parts that need to be avoided during the puncturing. The same approach could be easily adopted if the input is MRI.

Since in the proposed paper the PCNL procedure is addressed, the 3D model of the following parts are reconstructed: skin, ribs, kidney, calices, stones, ureter, aorta, inferior vena cava and all the neighboring structures (Figure 4.1(b)). The 3D reconstruction is performed starting from the DICOM images of the CT/MRI and segmenting the different parts on the basis of their Hounsfield Unit, which is the unit to measure the radiodensity into a CT/MRI.

Then, on the basis of the 3D reconstruction, a trajectory that reaches the target (i.e. the renal calix or the kidney stone) while avoiding the forbidden regions (green trajectory in Figure 4.1(c)) is generated and shown to the surgeon. Then, He/She can validate the trajectory or modify it simply by moving the inserted point: the trajectory will update accordingly.

Since the system will require reference points to perform a real-time localization of the 3D model on the body of the patient, three electrocardiogram (ECG) electrode patches following a L-shape directly on the back of the patient are applied before the execution of the initial CT scan. In the 3D reconstruction phase the contours of the ECG electrode patches are segmented and included in the 3D model and will be used as markers for the real-time registration (see Section 4.2.2).

4.2.2 Intra-Operative Phase

Once the pre-operative planning of the intervention is concluded, the 3D reconstruction is saved, and it will be used in an Augmented Reality (AR) application developed for an AR headset for visualizing the reconstruction directly on the body of the patient.

Since the CT/MRI scan is performed few days before the effective execution of the surgical intervention, a real-time registration is required to intra-operatively align the reconstructed model in the right position and orientation on the body of the patient. The real-time registration is performed on the basis of the three ECG electrode patches (Figure 4.1(d)). Details on the registration process will be provided in Section 4.2.2.

The surgeon, wearing the augmented reality headset (e.g. Microsoft HoloLens), will directly visualize the reconstructed model that, thanks to the previous registration, will be correctly located on the body of the patient (Figure 4.1(e)). Since the insertion trajectory is computed on the basis of the pre-operative CT/MRI scan, it may happen that the trajectory needs to be adjusted depending on the real location of the organs in the intra-operative phase. Thus, thanks to the AR application, the surgeon can directly adjust the insertion point in order to obtain a collision-free trajectory (i.e. a trajectory that still reaches the target while avoiding neighboring organs or ribs).

Once the optimal trajectory has been defined, the surgeon manually guides the robot from an initial position towards the desired trajectory for the percutaneous access. The robot is then used to gently assist the surgeon towards the trajectory. To this aim, the generation of virtual fixtures is included in the robot control scheme (Figure 4.1(f)).

Virtual fixtures are typically used in haptics and teleoperation architectures and they are generally defined as assistive forces applied to the haptic device that is used as the teleoperation master [58]. 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 the desired target, that can be either a specific operating point or a more complex geometric path. In this control architecture, the surgeon manually guides the robot through its end-effector. However, the concept of virtual fixtures can still be applied in order to generate virtual forces on the robot end-effector that can guide the needle (attached at the robot end-effector but handled by the surgeon's hand) towards the desired trajectory.

Since the proposed system have to assist a novel surgeon in reducing his/her learning curve, virtual fixtures are designed to intervene only when the surgeon is near to the desired trajectory, and thus for the final refinement of the position and the orientation

of the needle. The reason behind this choice is to have the surgeon trying to gain the right percutaneous access by himself/herself and performing the surgical intervention in autonomy but with the assistance of the proposed system that guarantees that the surgical operation is performed successfully without complications, especially at the early end of the learning curve.

Once the surgeon has moved the needle in the right position for the percutaneous access, with the right orientation, then he/she can perform the insertion. The surgeon is assisted by the virtual fixtures even during the needle insertion, in order to keep the needle along the right trajectory towards the target, while avoiding other anatomical structures.

The remaining part of this section provides the details on the three technical parts of the system: the real-time registration, the application for AR visualization, and the generation of assistive virtual fixtures.

Real-Time Registration

The reconstruction and the segmentation of the 3D patient model allow to define the reference system of the model itself on the basis of the L-shaped positioned triad of ECG electrode patches (Figure 4.1(d)). Indeed, the point corresponding to the center of the first patch is used to identify the origin of such reference system, while the line connecting the centers of the first and second patches determines the x-axis unit vector. Finally, the centers of the three patches define the xy-plane. Therefore, the origin and xyz-axis unit vectors of the 3D model reference system can be computed as follows:

$$\begin{aligned} \mathbf{O}_b &= \mathbf{P}_0 \\ \mathbf{v}_{b,x} &= \frac{(\mathbf{P}_1 - \mathbf{P}_0)}{||(\mathbf{P}_1 - \mathbf{P}_0)||} \\ \mathbf{v}_{b,z} &= \mathbf{v}_{b,x} \times \frac{(\mathbf{P}_2 - \mathbf{P}_0)}{||(\mathbf{P}_2 - \mathbf{P}_0)||} \\ \mathbf{v}_{b,y} &= \mathbf{v}_{b,z} \times \mathbf{v}_{b,x} \end{aligned} \tag{4.1}$$

where $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2 \in \mathbb{R}^3$ are the position of the centers of the patches, $\mathbf{O}_b \in \mathbb{R}^3$ and $\mathbf{v}_{b,x}, \mathbf{v}_{b,y}, \mathbf{v}_{b,z} \in \mathbb{R}^3$ (as column-vectors) are the origin and the unit vectors of the 3D model reference system. The symbol $\times$ indicates the cross product.

Once the 3D model reference system has been defined, the pose of each segmented anatomical part and the desired insertion trajectory, represented by an insertion point on the skin $\mathbf{P}_I \in \mathbb{R}^3$ and a target point on the kidney stone $\mathbf{P}_T \in \mathbb{R}^3$, can be mapped into to such a coordinate system (i.e. ${}^b\mathbf{P}_I, {}^b\mathbf{P}_T$, where the left apex indicates the reference system). It is worth noting that this procedure can be easily generalized to define a reference system by means of any triad of identified model points.

The real-time alignment of the reconstructed 3D model on the body of the patient is performed by localizing the three patches in the current spatial map of the environment, elaborated by the AR device. In particular, one Aruco marker [110] is placed on each patch, so that the centers of the patches coincide with the centers of the markers. By accessing the data coming from the stereo cameras available on the AR headset, the

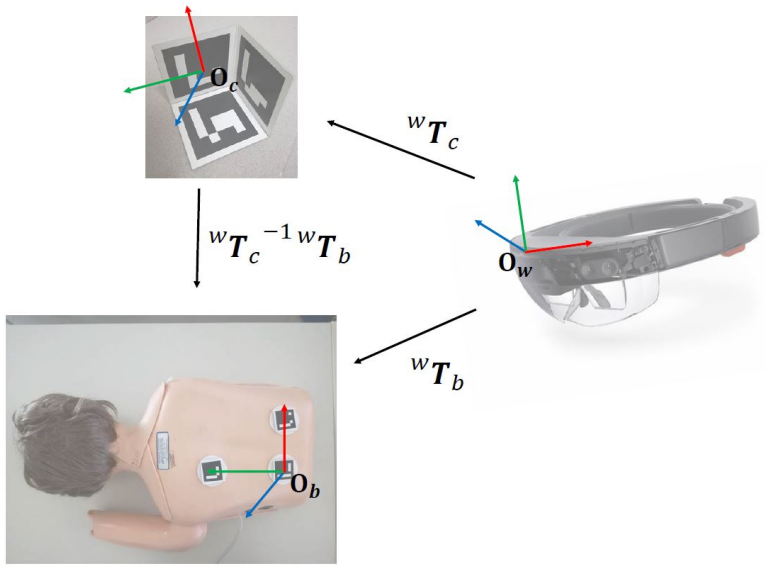


Figure 4.2: The reference systems and the transformation matrices among them.

unique id-code of the Aruco markers can be recognized, allowing the computation of the position of the pixels that contain the corners of the detected markers.

Assuming the knowledge of intrinsic and extrinsic camera parameters, each marker corner detected in the left camera frame can be triangulated with the same corner detected in the right camera frame, so that their 3D coordinates in the headset world reference system can be estimated. To improve the stability of the registration, a moving average filter on a window of 30 samples has been introduced on the spatial coordinates of all the marker corners detected by the AR device.

Then, the center of gravity (COG) of the four corners of each marker is kept to determine the central point of the three patches. Therefore, (4.1) can be used to compute the transformation matrix ${}^w\mathbf{T}_b$ of the patient body in the world reference system of the AR device:

$$ {}^w\mathbf{T}_b = \begin{bmatrix} \mathbf{v}_{b,x} & \mathbf{v}_{b,y} & \mathbf{v}_{b,z} & \mathbf{O}_b \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2} $$

Once the pose of the patient's body has been reconstructed, a further transformation has to be computed to express such pose in the robot coordinate system. For this purpose, a registration box (shown in the top-left of Figure 4.2) has been prepared, consisting of an open cube in which the three orthogonal internal faces contain an Aruco marker.

The box has to be fixed at a known pose with respect to the robot kinematics reference system and its faces have to be detected by the AR device so that a common reference system can be finally determined. Indeed, the corners of each marker are detected and triangulated, and the COG of the first marker corners is considered as the origin $\mathbf{O}_c$ of the cube reference system. Moreover, in this case, being the markers orthogonal to each other, each axis of the cube reference system (with unit vectors $\mathbf{v}_{c,x}$,$\mathbf{v}_{c,y}$,$\mathbf{v}_{c,z}$) can be estimated as the normal to the plane that best fits a set of marker corners [111].

Therefore, the transformation matrix ${}^{w}\mathbf{T}_{c}$ from the registration cube to the headset world reference system can be computed as:

$$ {}^{w}\mathbf{T}_{c} = \begin{bmatrix} \mathbf{v}_{c,x} & \mathbf{v}_{c,y} & \mathbf{v}_{c,z} & \mathbf{O}_{c} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3} $$

Figure 4.2 shows the AR headset world, the patient's body, and the cube reference systems.

Moreover, assuming the knowledge of ${}^{r}\mathbf{T}_{c}$, i.e. the matrix representing the transformation from the robot to the cube reference system, the insertion point and the target point can be transformed to the robot kinematics coordinates (${}^{r}\mathbf{P}_{I}$, ${}^{r}\mathbf{P}_{T}$) as follows:

$$ \begin{aligned} {}^{r}\mathbf{P}_{I} &= {}^{r}\mathbf{T}_{c} \cdot {}^{w}\mathbf{T}_{c}^{-1} \cdot {}^{w}\mathbf{T}_{b} \cdot {}^{b}\mathbf{P}_{I} \\ {}^{r}\mathbf{P}_{T} &= {}^{r}\mathbf{T}_{c} \cdot {}^{w}\mathbf{T}_{c}^{-1} \cdot {}^{w}\mathbf{T}_{b} \cdot {}^{b}\mathbf{P}_{T} \end{aligned} \tag{4.4} $$

Finally, the whole computational process of the AR device is summarized by Procedure 4.

Algorithm 4 Registration

1: **procedure** COMPUTEREGISTRATION(${}^{b}\mathbf{P}_{T}$,${}^{b}\mathbf{P}_{I}$,${}^{r}\mathbf{T}_{c}$)
2: Detect markers in the AR device and estimate their 3D position
3: Apply a moving average filter on the marker corners position
4: When all the registration cube markers are detected, compute ${}^{w}\mathbf{T}_{c}$
5: When all the markers are detected, compute their COG and ${}^{w}\mathbf{T}_{b}$ from (4.2)
6: Compute ${}^{r}\mathbf{P}_{T}$,${}^{r}\mathbf{P}_{I}$ from (4.4)
7: **end procedure**

Augmented Reality Visualization

The visualization of the AR scene is performed on the basis of the real-time detection of the Aruco markers and the resulting registration matrix ${}^{w}\mathbf{T}_{b}$. Indeed, as depicted in Figure 4.1(e), the hologram of the segmented 3D models representing the anatomical parts (ribs and organs) can be rendered overlapping the patient body. Moreover, the hologram of the insertion trajectory can be visualized in order to assist the surgeon in obtaining the right renal access.

Assistive Virtual Fixtures

The goal of the virtual fixtures is to assist the surgeon during the alignment and the insertion of the needle towards the desired trajectory for the right percutaneous access. In order to correctly perform the insertion of the needle, the instrument needs to be first aligned along the insertion direction and with a configuration such that it lies between the patient and the robot, outside the body. Once the instrument is aligned, the insertion can be executed.

Consider the insertion point on the skin ${}^r\mathbf{P}_I$ and the target point on the kidney stone ${}^r\mathbf{P}_T$ defined in Section 4.2.2, with respect to the robot base. The two points are computed in the pre-operative phase according to the 3D reconstruction and lie along the desired trajectory, selected as the one allowing to reach the kidney stone while avoiding other anatomical structures such as, e.g., the ribs. Consider now the insertion plane π, defined as the plane that passes through the insertion point $\mathbf{P}_I$ and that is normal to the insertion direction $\mathbf{P}_T - \mathbf{P}_I$:

$$\pi : \mathbf{ax} + \mathbf{by} + \mathbf{cz} + \mathbf{d} = 0 \tag{4.5}$$

where the parameters $\mathbf{a}, \mathbf{b}$, and $\mathbf{c}$ are the components of the insertion direction unit vector, while $\mathbf{d}$ can be easily computed by substituting the components of the insertion point $\mathbf{P}_I$ into the plane Equation (4.5).

Considering the plane π, as defined in (4.5) and the insertion direction $\mathbf{P}_T - \mathbf{P}_I$ the insertion frame ${}^r\mathbf{T}_I$, i.e. the pose the instrument should have when the needle starts perforating the skin, where r is the robot base reference frame and I is the insertion point, can be computed as follows: 1. the z-axis is given by the insertion direction unit vector, 2. the x-axis can be computed considering the unit vector pointing to an arbitrary point on the insertion plane and, 3. the y-axis can be computed as the cross product of the z-axis and the x-axis.

Once computed the insertion frame ${}^r\mathbf{T}_I$, the target frame on the kidney stone ${}^r\mathbf{T}_T$ and the alignment frame ${}^r\mathbf{T}_A$ can be calculated. The target frame of the puncturing ${}^r\mathbf{T}_T$ represents the pose of the kidney stone, and it can be computed exploiting the distance between the target point $\mathbf{P}_T$ and the insertion point $\mathbf{P}_I$. The alignment frame ${}^r\mathbf{T}_A$ represents the desired instrument pose at the beginning of the insertion. This frame is computed applying a safety distance d_s to the insertion frame ${}^r\mathbf{T}_I$ and it is computed such that it lies between the patient and the robot, outside the body.

Figure 4.3 shows an example of computation of the three main frames starting from the target point $\mathbf{P}_T$ and the insertion point $\mathbf{P}_I$.

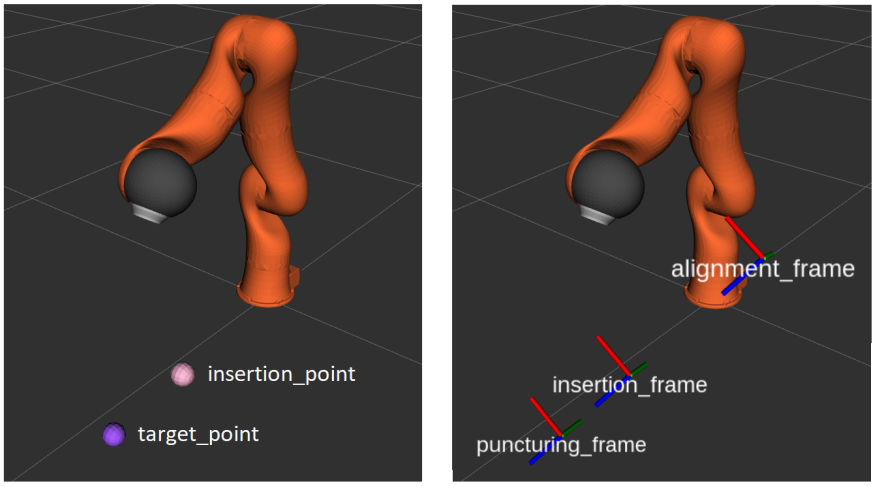


Figure 4.3: The computation of the main Cartesian coordinate system starting from the target point on the kidney stone (purple sphere) and the insertion point on the skin (pink sphere).

The procedure 5 summarizes the processing steps that allow to obtain the frames ${}^r\mathbf{T}_T$,${}^r\mathbf{T}_I$, ${}^r\mathbf{T}_A$.

Algorithm 5 Frame Computation

1: **procedure** COMPUTEFRAMES(${}^r\mathbf{P}_T$,${}^r\mathbf{P}_I$,${}^r\mathbf{T}_c$)
2: Compute the parameters $\mathbf{a}, \mathbf{b}$,$\mathbf{c}$,$\mathbf{d}$ of the insertion plane π from (4.5)
3: Fix the orientation of the frames considering the z-axis normal to π
4: Set the insertion frame ${}^r\mathbf{T}_I$
5: Set the target frame on kidney stone ${}^r\mathbf{T}_T$
6: Set the alignment frame ${}^r\mathbf{T}_A$ applying the safety distance d_s
7: **end procedure**

In order to generate the virtual fixtures, the execution of the percutaneous renal access for PCNL is split into three different phases:

- *Alignment*: the robot is in the initial configuration and must guide the surgeon towards the alignment frame ${}^r\mathbf{T}_A$.
- *Stiffening*: the alignment frame ${}^r\mathbf{T}_A$ is reached. The robot has to change its stiffness in order to allow the surgeon to move the instrument along the insertion direction and not outside the desired trajectory.
- *Guidance*: the desired stiffness is reached. The surgeon can now move the instrument along the insertion direction to perform the needle insertion.

The virtual fixtures are computed in all the three phases as follows:

$$\begin{aligned} \mathbf{F_e} &= \mathbf{k_F}\Delta\mathbf{x_e} - \mathbf{b_F}\dot{\mathbf{x}}_\mathbf{e} \\ \boldsymbol{\tau}_r &= \mathbf{k_T}\Delta\theta_\mathbf{b} - \mathbf{b_T}\dot{\theta}_\mathbf{b} \end{aligned} \tag{4.6}$$

where $\mathbf{F_e} \in \mathbb{R}^3$ is the commanded force with respect to the robot end-effector, $\mathbf{k_F}, \mathbf{b_F} \in \mathbb{R}^3$ are linear spring-damper coefficients, $\Delta\mathbf{x_e} \in \mathbb{R}^3$ is the position error between the desired frame and the robot frame with respect to robot end-effector and $\dot{\mathbf{x}}_\mathbf{e} \in \mathbb{R}^3$ is the linear velocity of the robot end-effector with respect to robot end-effector. $\boldsymbol{\tau}_r \in \mathbb{R}^3$ is the commanded torque with respect to the robot base, $\mathbf{k_T}, \mathbf{b_T} \in \mathbb{R}^3$ are rotational spring-damper coefficients, $\Delta\theta_\mathbf{b} \in \mathbb{R}^3$ is the rotational error between the desired frame and the robot frame with respect to the robot base, and $\dot{\theta}_\mathbf{b} \in \mathbb{R}^3$ is the angular velocity of the robot end-effector with respect to the robot base.

During the alignment phase, the desired frame is imposed to be the alignment frame ${}^r\mathbf{T}_A$. In this phase, if the initial configuration of the robot is far from the alignment frame, the position errors $\mathbf{\Delta x_e}$ and the rotational errors $\Delta\theta_\mathbf{b}$ could be very high. In order to avoid excessive forces and torques, the linear spring coefficients $\mathbf{k_F}$ and rotational spring coefficients $\mathbf{k_T}$ are set as follow:

$$\begin{aligned} \mathbf{k_F} &= \mathbf{k_F^{min}} + \left(1 - e^{\frac{||\Delta\mathbf{x_e}||}{\tau_F}}\right)\left(\mathbf{k_F^{max}} - \mathbf{k_F^{min}}\right) \\ \mathbf{k_T} &= \mathbf{k_T^{min}} + \left(1 - e^{\frac{||\Delta\theta_\mathbf{b}||}{\tau_T}}\right)\left(\mathbf{k_T^{max}} - \mathbf{k_T^{min}}\right) \end{aligned} \tag{4.7}$$

where $\mathbf{k_F^{min}}, \mathbf{k_F^{max}}, \mathbf{k_T^{min}}, \mathbf{k_T^{max}}$ are the minimum and the maximum value of the linear spring coefficients and the rotational spring coefficients, respectively, and τ_F, τ_T are user-

defined parameter used to define the transition of the coefficient between the limit values.

The idea is that for high errors the forces and the torques should be low, in order to allow the surgeon to freely move the robot. As the error decrease, the surgeon is increasingly being attracted to the desired configuration. Thanks to this solution, virtual fixtures intervene only when the surgeon is near to the desired trajectory, and thus for the final refinement of the position and the orientation of the needle. The surgeon can try to gain the right percutaneous access by himself/herself in autonomy but the assistance of the proposed system guarantees that the surgical operation is performed successfully without complications.

Once the alignment frame is reached, the robot changes its stiffness to the desired value $\mathbf{k_F^g}, \mathbf{k_T^g}$ in the desired time t_s. This stiffening allows the robot to increase the accuracy with which the insertion trajectory is followed and allows the surgeon to start the assisted insertion of the needle. In particular, the stiffness will be low along the trajectory direction, while it will be high in the other components, in order to prevent the user from exiting the desired trajectory.

In the guidance state, the desired frame is set to be the target frame ${}^r\mathbf{T}_T$ and the spring coefficients are kept constant to the values of the stiffening phase $\mathbf{k_F^g}, \mathbf{k_T^g}$. In order to allow the surgeon to move the instrument along the insertion trajectory, the value of the linear spring coefficient along the z-axis is set equal to 0.

4.3 Validation

The experimental validation is described in the following and it is organized into three main phases:

- *Registration*: the accuracy of the location of the 3D reconstruction of the anatomical model of the patient with respect to the real body is evaluated.
- *Performances*: a comparison between the performances of the system with and without the use of the virtual fixtures is reported.
- *Usability*: a validation on a sample of 11 users is performed and a statistical analysis of the usability of the system is reported.

The experimental setup included a KUKA LWR 4+ 7-DOF robot able to return the virtual fixtures to the user. The surgical scenario is reproduced using a 3D printed handle with a needle, mounted on the end-effector of the robot. The AR headset being used is the Microsoft HoloLens that is worn by the user to display the 3D reconstruction of the body of the patient and to perform the real-time registration. The 3D reconstruction is visualized on a manikin lying down on the table where the robot is fixed. Once the registration is accomplished, the user is free to handle the surgical tool and perform the procedure, following the phases described in Section 4.2.2: *alignment*, *stiffening*, and *insertion*.

In the experiments in which the virtual fixtures are turned off, the procedure is executed manually, with the robot gravity-compensated and using as feedback the only visual information provided by the HoloLens. These experiments will be hereafter referred to as *manual experiments*. In the experiments in which the virtual fixtures are turned on, the user still has the visual information but the movements are assisted by the virtual fixtures generated through the robot, following the description reported in 4.2.2. These experiments will be hereafter referred to as *assisted experiments*.

In the *manual experiments*, the *stiffening* phase is skipped. The experiments were conducted by 11 users (4 female, 7 male, from 24 to 32 years old) performing both the *manual experiment* and the *assisted experiment*. Moreover, the users were asked to fill a questionnaire, in order to obtain a qualitative evaluation of the user's perspective.

4.3.1 Registration

In order to evaluate the quality of the registration, the transformation matrix ${}^{r}\mathbf{T}_{b}$ of the patient body in the robot reference system provided by the HoloLens is compared with the one computed executing the same registration algorithm but starting from the position of the centers of the patches measured using the robot. The linear and angular error norm between the two transformation matrices resulted to be 15.80 mm and 4.12 deg, respectively.

The angular error is computed as the absolute value of the angle in the axis-angle representation of the following rotation matrix:

$$ {}^{r}\mathbf{R}_{b}(h)^{T}\ {}^{r}\mathbf{R}_{b}(r) \tag{4.8} $$

where ${}^{r}\mathbf{R}_{b}(h)$ is the transformation matrix of the patient body in the robot reference system provided by the HoloLens while ${}^{r}\mathbf{R}_{b}(r)$ is the transformation matrix of the patient body in the robot reference system computed using the robot.

The resulting angular error is quite contained while the linear error is more relevant. This is due to the fact that the orientation estimation is mediated by the use of all the markers, while the linear strongly depends on the ability of the HoloLens camera to recognize the position of the single marker. As will be described in Section 4.4, this error can be strongly mitigated by considering additional information, such as merging the HoloLens data with the information acquired in real-time through an ultrasound probe or through an external tracking system.

4.3.2 Performances

The performances of the system are evaluated by comparing the results of the *manual experiments* and the ones from the *assisted experiments*. Figures 4.4 and 4.4 reports the evolution over time of the linear and angular error norms between the desired position of the instrument and its real position. In particular, Figure 4.4 shows the values of the errors during the *alignment phase*, where the target is the alignment frame ${}^{r}\mathbf{T}_{A}$,

Figures 4.5 shows the values of the errors during the *insertion phase*, where the target is the target frame ${}^{r}\mathbf{T}_{T}$. The *stiffening phase* is omitted since it takes place only for the *assisted experiment*. The plots refer to the experiments of one random user.

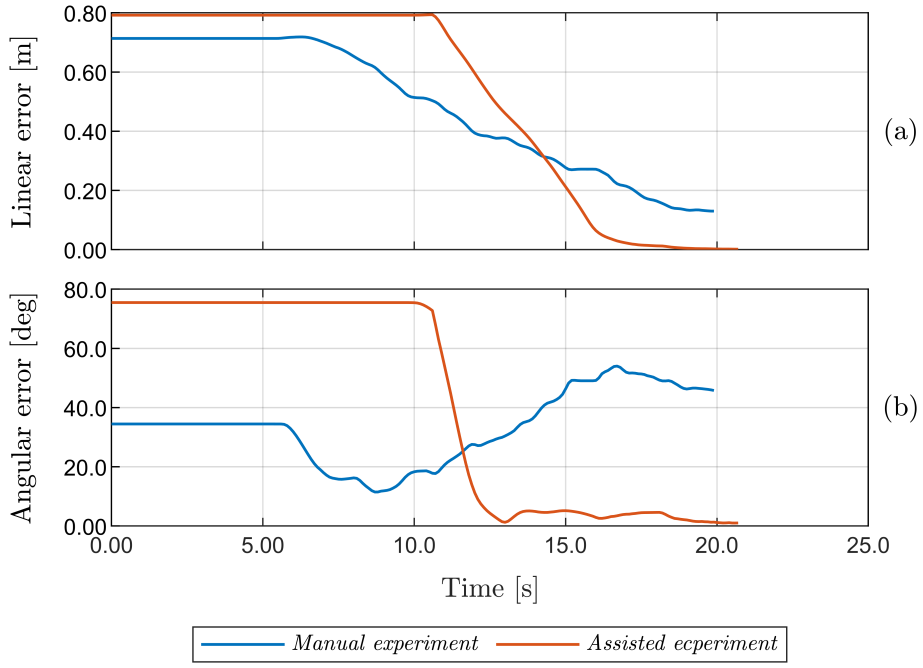


Figure 4.4: Linear and angular error norms during the *alignment phase*.

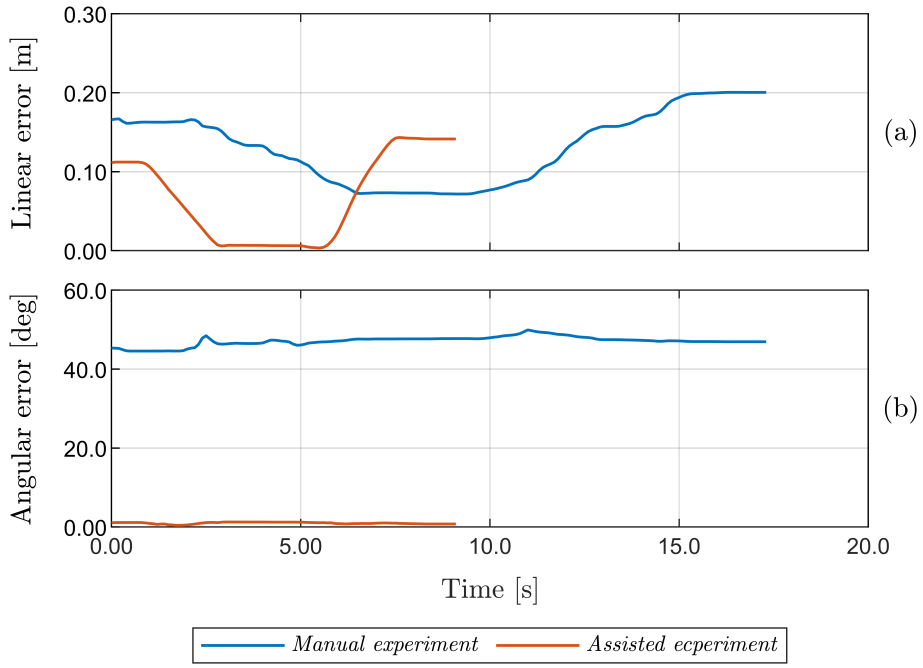


Figure 4.5: Linear and angular error norms during the *insertion phase*.

The angular error norm is computed considering the angle between the desired direction and the real one, and it can be calculated as:

$$|| \arccos(\bar{\mathbf{z}}_{\mathbf{m}} \cdot \bar{\mathbf{z}}_{\mathbf{d}}) || \tag{4.9}$$

where $\bar{\mathbf{z}}_{\mathbf{m}}$ and $\bar{\mathbf{z}}_{\mathbf{d}}$ denotes the measured and the desired z-axis unit vector of the robot end-effector, and the symbol $\cdot$ it is used to indicate the dot product.

From the results shown in Figures 4.4 and 4.5, it can be observed that the execution time for the *assisted experiment* is smaller or, at least, comparable to the execution time of the *manual experiment*. However, the errors for the *assisted experiment* are clearly less than the ones of the *manual experiment*.

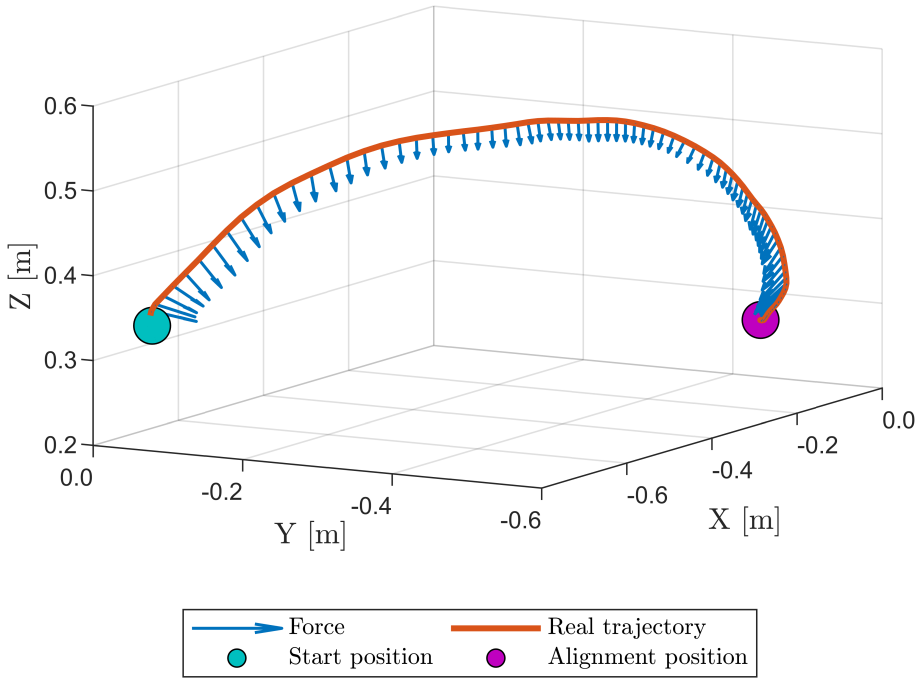


Figure 4.6: Forces and trajectory during the *alignment phase*

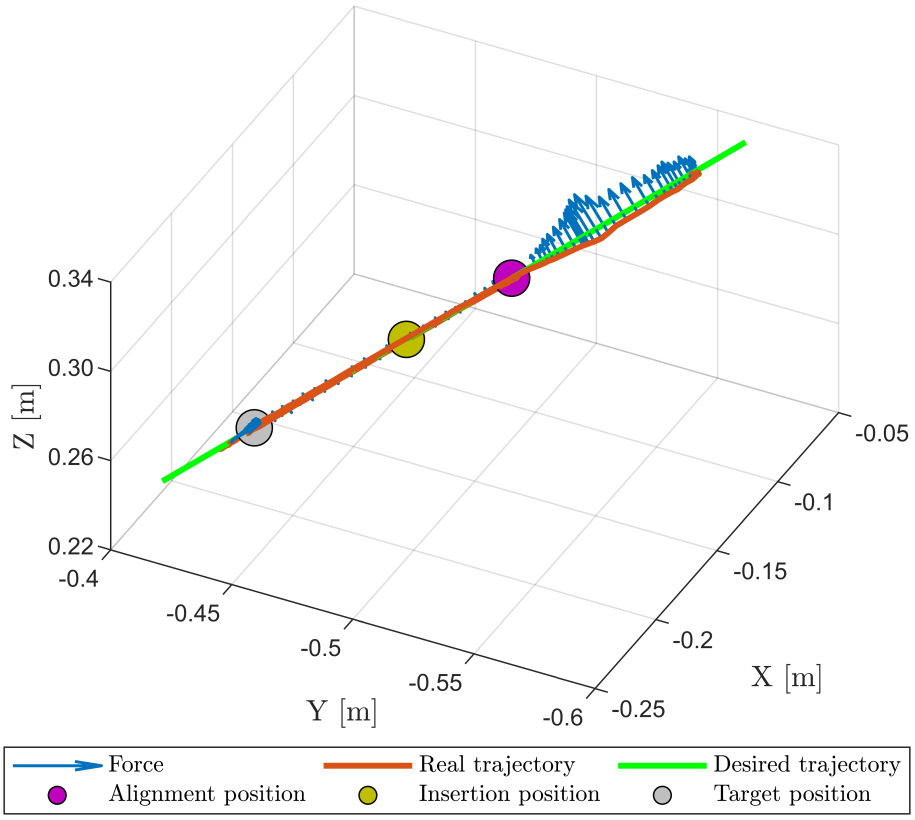


Figure 4.7: Forces and trajectory during the *insertion phase*

Figures 4.6 and 4.7 reports the trajectory and the forces of/on the instrument during the *assisted experiment*. During the *alignment phase*, reported in Figure 4.6, the forces

attract the robot towards the alignment position. However, the user can still correct the robot trajectory. This can be seen looking at the forces that push to the alignment position but the movement follows another trajectory. This behavior is due to the user, that moves the instrument along a different trajectory to avoid a collision with the patient body. During the *insertion phase*, reported in Figure 4.7, the virtual fixtures keep the robot along the insertion direction and do not allow the tip of the instrument to go beyond the target. This behavior can be observed 1. near the target position, where the forces push the instrument to go back, and 2. outside the insertion position, where the user intentionally tried to take the robot out of the insertion direction but the fixtures take the instrument on the right trajectory.

In order to evaluate the improvements in the performances when the procedure is executed using the virtual fixtures, a statistical analysis over the following metrics is conducted: 1. the execution time, 2. the linear error at the target, and 3. the angular error at the target.

Table 4.1 reports the mean values and the standard deviation values of the described metrics for both the *manual experiments* and the *assisted experiments*. These results confirm what observed in Figures 4.4 and 4.5 and allows to state the positive contribution of the virtual fixtures when performing the procedure.

	Manual exp.		Assisted exp.	
	mean	std. dev.	mean	std. dev.
Execution time [s]	25.89	14.90	19.56	4.59
Linear error [mm]	37.93	24.15	15.18	12.75
Angular error [deg]	30.12	31.43	1.50	0.61

Table 4.1: Performance statistical results comparing the use of virtual fixtures with respect to the manual execution of the procedure.

4.3.3 Usability

The effort required to use a system or a device defines its usability. In order to state that the use of the virtual fixtures makes the system more usable with respect to the execution of the procedure with only the AR support, a statistical analysis over the NASA Task Load Index (NASA-TLX) [112] is also conducted.

The NASA-TLX rates the perceived workload when assessing a task and it is calculated on the basis of a questionnaire which is filled in by the users after having performed the task. In particular, the raw TLX value is computed since it has been shown that it might increase the experimental validity. Table 4.2 reports the mean values and the standard deviation values of the NASA-TLX for both the *manual experiments* and the *assisted experiments*.

It can be observed that both the mean value and the standard deviation of the NASA-TLX for the *assisted experiments* are less than the ones obtained for the *manual experiments*. This allows to state that the virtual fixtures return a positive contribution

in terms of usability of the system and not only from the performance point of view if compared to the application of a sole AR support.

	Manual experiments		Assisted experiments	
	mean	std. dev.	mean	std. dev.
NASA-TLX	55.30	14.04	28.64	14.53

Table 4.2: NASA Task Load Index statistical results comparing the use of virtual fixtures with respect to the manul execution of the procedure.

4.4 Conclusions

In this work, an innovative solution, based on an AR application combined with a robotic system, that can assist both an expert surgeon in improving the performance of the surgical operation and a novel surgeon in strongly reducing his/her learning curve is developed.

The work addressed percutaneous nephrolithotomy (PCNL) as a use case, since it is considered to be the gold standard for the treatment of patients with renal stones. However, the proposed system architecture can be easily adopted in all the surgical procedures that require pre-operative planning and intra-operative navigation for gaining access to a specific target, especially in cases where the intervention learning curve for novel surgeons is very steep.

Future works aim at improving the registration procedure by including information acquired in real-time through an ultrasound probe. Moreover, the overall system will be evaluated and validated with a larger sample of surgeons that will perform PCNL on phantoms that emulate the human abdomen.

Chapter 5

Dynamic based Remote Centre of Motion Control

Chapter 5 reports the development of a torque control strategy for the implementation of virtual RCM on light-weight torque controlled manipulators. The dynamic model of the manipulator subject to the RCM is formulated and exploited to synthesized the control action, leaving the user the possibility to freely design the manipulator behavior. The results obtained after a first validation of the controller are extended in order to improve the capabilities of the controller to implement compliance motion control and make the robot effectively interact with the environment. The controller is then validated on a simulated setup and then experimentally on a real torque-controlled manipulator. Finally, the controller is extended to admittance controlled manipulator and experimentally validated on a teleoperation setup.

The work presented in this chapter is published in [14, 8].

5.1 Introduction

In RAMIS procedures, the insertion of the surgical tools through the incision on the patient's abdominal wall constrains the DOFs of the tools to be only rotations around the penetration point and a translation along the tool axis. The resultant tool's motion from these constraints is called the remote center of motion RCM.

With respect to standard Minimally Invasive Surgery, where the tools are moved manually by the surgeon, in RAMIS the tools are moved by the surgical robot which has to guarantee the RCM constraint and the implementation of the desired motion.

Different strategies for solving the RCM motion control problem have been proposed in the literature: mechanically constrained RCM, passive joint, and programmable RCM [113].

Mechanically constrained RCM consists of building a mechanical robotic structure such that the system can move only compliantly with the RCM. These strategies have the advantage of increasing safety since the RCM constraint is mechanically guaranteed. Moreover, a simple controller is required. The da Vinci®[25], the RTW [114] and the ARTEMIS [115] are just a few examples of robotic system for RAMIS that uses a mechanical constrained RCM. However, these systems typically require additional DOFs for the manipulator base to accurately position the mechanical RCM to the penetration point [116].

Passive joint strategies exploit a passive joint to make the surgical tool pivoting in two DOFs allowing the tool to automatically align with respect to the RCM, guaranteeing safety even if the patient moves during the operation. For example, the Zeus surgical system was developed with this kind of strategy [117]. The main drawback of this strategy is the low precision due to the backlash of the joints.

Programmable RCM strategies rely on a dedicated controller which takes care of implementing the desired motion while guaranteeing the RCM constraint. This kind of strategy is often designed for standard manipulators, typically redundant with respect to the required 4 DOFs. Indeed, redundancy can be exploited to perform different tasks simultaneously [118], such as maintaining the RCM while completing the surgical procedure. These strategies have the advantage of being less expensive and more flexible since the position of the RCM can be reconfigured online. The DLR MIRO robot is a clear example of a robot that implements a programmable RCM [119].

Different strategies have been developed in the literature for implementing programmable RCM control, both in the task space [120] and in the joint space [121].

Hierarchical control strategies [122, 123] have been proven to be very efficient when multiple tasks need to be executed at the same time. These strategies have also been extended to programmable RCM control. In [124] a 3D path following problem is solved together with RCM constraint. A similar approach has been developed in [125] for a trajectory tracking problem.

A different strategy combining hierarchical control and the definition of an RCM Jacobian matrix is proposed in [126], where a PD control action enforce the RCM constraint while executing secondary tasks. The RCM Jacobian matrix has been also exploited in [127] where a closed-loop inverse kinematic controller is developed to guarantee the RCM constraint while tracking the desired trajectory.

Other strategies like [128, 129] exploit optimization processes in order to compute RCM compliant joint velocities in real-time. However, unfeasible solutions and unpredictable computational time introduced by the optimization process can negatively affect the control performance.

The strategies presented so far model the RCM constraint from a purely kinematic point of view, disregarding the effects the constraint produces on the dynamics of the robot. This is fine as long as position or velocity controlled robots are considered for the surgical tasks since the low-level controllers compensate the dynamics of the robot. Nevertheless, lightweight torque-controlled robots allow achieving superior performance in terms of achievable precision and of control of the interaction between the robot and the environment [130, 131]). These features are becoming more and more relevant in the surgical scenario, where robots can autonomously execute some surgical tasks [83] or assist the surgeon [132].

A torque controller was proposed in [126], but the dynamic of the overall constrained system was not exploited, compensating the dynamic effect using feedback control actions.

This work stems from the experience acquired during the entire saras project and proposes a controller that allows the use of any torque-controlled manipulator for RAMIS procedures. In the specific case of the SARAS project, this controller proposes a low-cost alternative to the SARAS arms. Indeed, all SARAS platforms require two robotic arms holding laparoscopic tools and able to comply with the RCM constraint.

The development of the overall architecture was split into three phases:

- a first architecture was designed exploiting closed chain manipulators theory, and synthesizing the controller considering the effects the RCM constraint produces at a dynamic level. The overall system emulates a virtual dynamics that enforces the RCM constraint, and, for which, any desired behavior can be designed and reproduced by means of the torque controller.
- an improved and optimized architecture was later developed and validated, addressing the main problems arose during the development and validation of the previous architecture.
- the improved controller was validated in a bilateral SMSS teleoperation set-up, demonstrating its usefulness even for non-autonomous applications.

5.2 Problem Statement

Consider a n-DOFs manipulator represented by the following Euler-Lagrange model:

$$M\big(q(t)\big)\ddot{q}(t) + C\big(q(t), \dot{q}(t)\big)\dot{q}(t) + G\big(q(t)\big) = \tau(t) + \tau^{ext}(t) \tag{5.1}$$

where $q(t) \in \mathbb{R}^n$ is the generalized coordinates vector, $M\big(q(t)\big) \in \mathbb{R}^{n\times n}$ is the inertia matrix, $C\big(q(t), \dot{q}(t)\big) \in \mathbb{R}^{n\times n}$ is the centrifugal force and Coriolis forces matrix and $G\big(q(t)\big) \in \mathbb{R}^n$ is the gravitational vector. The term $\tau(t) \in \mathbb{R}^n$ is the joint torque vector and $\tau^{ext}(t)$ is the vector of generalized external torques, i.e. the torques applied by the by the environment.

For sake of clarity, the time-dependence of the variables will be hereafter omitted when the context is clear.

The end-effector of the manipulator is endowed with a laparoscopic tool that is inserted inside the patient's abdominal wall. As already mentioned, this insertion constrains the movement of the tool: three rotations about the insertion point and a translation along the tool direction are allowed in order to guarantee the RCM.

The goal is to build a torque controller for (5.1), i.e. design τ in (5.1), that dynamically constrains the motion of the tool according to the RCM constraints and that, at the same time, allows to reproduce a desired dynamic behavior.

5.3 RCM Controller V1.0

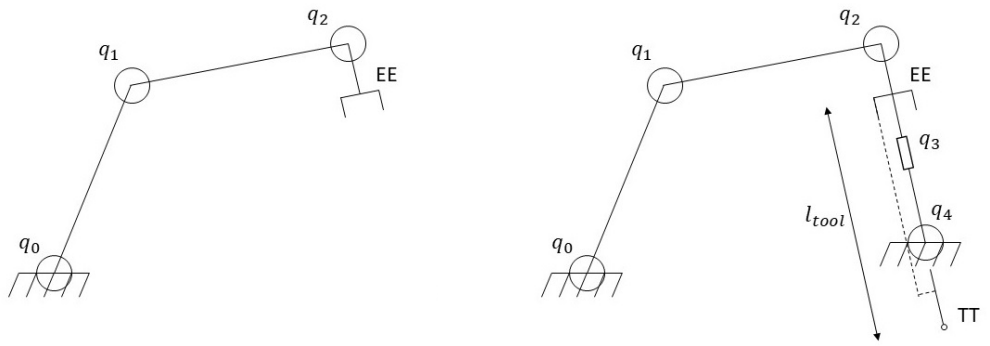
Before moving to the description of the controller, it is necessary to show how the RCM constraint is modeled.

In order to graphically convey the basic idea by which this modeling is performed, consider a simple example with a 3 DOFs planar manipulator, as shown in Figure 5.1a. In the planar scenario, the RCM constraint limits the number of DOFs of the surgical tool from three to two: one rotation around the insertion point and a translation along the tool direction. This implies that the constrained model can be obtained from the 3 DOFs planar manipulator shown in Figure 5.1a by adding two additional virtual joints. Figure 5.1b shows the augmented model. In particular:

- the prismatic virtual joint q_3 emulates the movement of the tool along its direction.
- the rotational virtual joint q_4, fixed on the insertion point, emulates the presence of the RCM, allowing only rotation of the tool around the insertion point.

The laparoscopic tool is fixed on the end-effector of the manipulator, and its direction is always coincident with the first virtual joint, i.e. the linear guide. In this way, the tool passes always through the RCM.

This procedure can be simply extended to the 6-DOFs case by replacing the joint q_4 with a spherical joint, and considering a manipulator with at least 6 DOFs.



(a) **(b)**

Figure 5.1: RCM modeling example for a 3-DOFs planar manipulator. (TT) Tool tip. (l_{tool}) Tool length.

It is worth noting that modeling the system in this way leads to a virtually closed chain manipulator since both the first and the last joints are fixed. From hereafter, the manipulator (5.1) will be referred to as RM (real manipulator), and the virtually closed chain manipulator as VM (virtual manipulator). In the planar example reported in Figure 5.1, RM is the manipulator in Figure 5.1a and VM is the manipulator in Figure 5.1b.

The desired behavior is designed for VM at the torques level, accordingly with the task to be performed. Given the desired behavior, the dynamics of VM is first emulated. The emulation produces the accelerations of VM, which are used as the reference for RM, which implements the real task. Based on this, the control problem can be split into two main problems:

- emulate the dynamics of VM and compute the reference accelerations.
- track the acceleration reference with a torque controller.

For each problem, a dedicated controller is developed. The *outer controller* emulates the dynamics of VM using the information collected from RM and applying the desired torque. The emulation provides the desired accelerations, that are driven to the *inner controller*, whose role is to make RM track the acceleration reference.

5.3.1 Outer Controller

From the joint positions and velocities of RM, the *outer controller* computes the joint positions and velocities of VM by considering the virtual joints, as described in Section 5.2 (see Section 5.3.3 for more details).

VM is a dynamic system with $m = n + 4$ joints, where $n \geq 6$ is the number of DOFs of RM. The dynamic model of VM can be obtained starting from its Euler-Lagrange model:

$$M_c(q_c)\ddot{q}_c + C_c(q_c, \dot{q}_c)\dot{q}_c + G_c(q_c) = \tau_c \tag{5.2}$$

where $q_c \in \mathbb{R}^m$ is the generalized coordinates vector, $M_c(q_c) \in \mathbb{R}^{m \times m}$ is the inertia matrix, $C_c(q_c, \dot{q}_c) \in \mathbb{R}^{m \times m}$ is the centrifugal force and Coriolis forces matrix, $G_c(q_c) \in \mathbb{R}^m$

is the gravitational vector and $\tau_c \in \mathbb{R}^m$ is the joint torque vector.

Since the *outer controller* is used as a dynamics emulator, the dynamic parameters that characterize the model (e.g. the inertia properties of the links) can be freely chosen by the user, while the kinematic quantities (e.g. links length) depend on RM.

The model in (5.2) is not subject to any constraint, namely, applying a generic set of torques τ_c does not guarantee that the system will behave like a closed chain system since the ending position of the chain is not fixed.

Assume that the system in (5.2) is subject to a certain general set of kinematic constraint, e.g. the chain has to be closed. The constraint can be represented as:

$$\psi_i(q_c) = 0 \quad i = 1, ..., p \tag{5.3}$$

where $\psi_i(q_c)$ represents the generic i-th constraint equation and p is the number of constraints equations.

Differentiating Equation (5.3) with respect to time twice yields the following constraint equations:

$$\ddot{\psi}_i(q_c, \dot{q}_c, \ddot{q}_c) = \frac{\partial^2 \psi_i}{\partial^2 q_c}\dot{q}_c^2 + \frac{\partial \psi_i}{\partial q_c}\ddot{q}_c = 0 \quad i = 1, ..., p \tag{5.4}$$

By setting:

$$A_i(q_c) = \frac{\partial \psi_i}{\partial q_c} \quad b_i(q_c, \dot{q}_c) = \frac{\partial^2 \psi_i}{\partial^2 q_c}\dot{q}_c^2 \tag{5.5}$$

Equation (5.4) can be rewritten as:

$$A_i(q_c)\ddot{q}_c = b_i(q_c, \dot{q}_c) \tag{5.6}$$

This means that by setting:

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_p \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \end{bmatrix} \tag{5.7}$$

the set of constraints (5.3) can be thus rewritten compactly, as:

$$A(q_c)\ddot{q}_c = b(q_c, \dot{q}_c) \tag{5.8}$$

Using (5.8) in (5.2), the dynamic model of the system in (5.2) subject to the set of constraints (5.3) can be rewritten as:

$$M_c(q_c)\ddot{q}_c = Q(q_c, \dot{q}_c) + Q_{constr}(q_c, \dot{q}_c) \tag{5.9}$$

where Q can be obtained from (5.2) as:

$$Q(q_c, \dot{q}_c) = \tau_c - C_c(q_c, \dot{q}_c)\dot{q}_c - G_c(q_c) \tag{5.10}$$

and Q_{constr} is the additional joint torque vector that arises to ensure that the constraint

requirements (5.3) are satisfied, and it can be explicitly obtained using the Udwadia-Kalaba Equation [133] as:

$$Q_{constr}(q_c, \dot{q}_c) = M_c(q_c)^{\frac{1}{2}} B^{+}(q_c)\Big(b(q_c, \dot{q}_c) - A(q_c, \dot{q}_c)M(q_c)^{-1}Q(q_c, \dot{q}_c)\Big) \tag{5.11}$$

in which:

$$B(q_c) = A(q_c, \dot{q}_c)M_c(q_c)^{\frac{1}{2}} \tag{5.12}$$

and the superscript + represents the Moore-Penrose generalized inverse.

Let $x_c(q_c) \in \mathbb{R}^3$ be the position of the last joint of the model in (5.2), i.e. the position of the virtual spherical joint that emulates the presence of the RCM. From a kinematic point of view, constraining the serial chain to be close can be achieved forcing that:

$$\psi(q_c) = x_c(q_c) - x_{rcm}^{des} = 0 \tag{5.13}$$

where $x_{des}^{rcm} \in \mathbb{R}^3$ is the desired position at which the chain has to close, i.e. the position of the RCM.

This means that $A_i(q_c, \dot{q}_c)$ and $b_i(q_c, \dot{q}_c)$ can be computed by differentiating each component of (5.13) with respect to time twice. The overall constraint matrices can be then computed by grouping A_i and b_i as in (5.7). VM can be then modeled substituting them in (5.9) using (5.10), (5.11), and (5.12).

The term τ_c in (5.2) can be chosen to implement any desired behavior of VM. Once τ_c has been designed, the accelerations of VM can be simply computed by:

$$\ddot{q}_c = M_c(q_c)^{-1}\big(Q(q_c, \dot{q}_c) + Q_{constr}(q_c, \dot{q}_c)\big) \tag{5.14}$$

As observable from (5.9), the maintenance of the constraint is independent of the choice of τ_c. This, in association with the free choice of the dynamic parameters of the model, gives an extreme flexibility to the system. In fact, it is possible to design the desired behavior of (5.2) disregarding the RCM constraint and then, exploiting (5.14), achieving the acceleration corresponding to the RCM compliant (i.e. it satisfies the RCM constraint) version of the desired behavior.

In order to make RM behave like VM, it is necessary to build a controller that enables RM in (5.1) to track the acceleration (5.14). The design of the controller will be shown in the next subsection.

Since the joints position and velocity of VM are computed starting from the joints position and velocity of RM, VM and RM will be always synchronized in terms of positions and velocities. This means that tracking the accelerations in (5.14) will allow RM to behave like VM.

5.3.2 Inner Controller

The goal of the *inner controller* is to guarantee that RM as in (5.1) follows the acceleration set-point provided by the *outer controller*.

The acceleration computed by *outer controller* $\ddot{q}_c$ as in (5.14) lies in $\mathbb{R}^m$ while the acceleration of RM $\ddot{q}$ lies in $\mathbb{R}^n$ with $m = n + 4$. Let define $\ddot{q}^{\,des} \in \mathbb{R}^n$ as the first n components of $\ddot{q}_c$, the ones that correspond to real controllable joints:

$$\ddot{q}^{\,des} = \begin{bmatrix} I_n & 0_{n\times 4} \end{bmatrix} \ddot{q}_c \tag{5.15}$$

where I_n and $0_{n\times 4}$ are the identity matrix and the zero matrix of proper dimensions, respectively.

The problem of following $\ddot{q}^{\,des}$ can be achieved design a controller such that:

$$\tau = M\ddot{q}^{\,des} + C(q,\dot{q})\dot{q} + G(q) - \tau^{ext} + \tau^{rcm} \tag{5.16}$$

where $\tau^{rcm} \in \mathbb{R}^n$ is a torque contribution introduced to improve the RCM position tracking.

From a theoretical point of view, the term τ^{rcm} is not needed since substituting (5.16) into (5.1) setting $\tau^{rcm} = 0$ brings to:

$$\ddot{q} = \ddot{q}^{\,des} \tag{5.17}$$

which means perfect acceleration tracking.

The RCM constraint introduced with (5.13) in (5.8) and then in (5.9) for VM acts on the acceleration level $\ddot{q}_c$, and depends on the positions q_c and velocities $\dot{q}_c$. Misalignment in terms of q_c and $\dot{q}_c$ will occur when the integration time of the model is finite (e.g. discrete time controller). These misalignments cause errors in evaluating the constraint function (5.8), consequently, in the evaluation of the model (5.9) and, finally, in the evaluation of the acceleration (5.14), rapidly degrading the overall performance.

As reported in the following, the term τ^{rcm} enforces the tracking of the RCM position, improving the evaluation of the constraint (5.13) and (5.8). This allows to increase the performance of the emulation of VM and the fidelity of the acceleration reference $\ddot{q}_c$, increasing the performance of the overall system.

A strategy to define the term τ^{rcm} is to design a feedback controller and monitor the position and the speed of the RCM during the motion of the system. As proposed in [126], the computation of the position and of the speed of the RCM can be performed as in the following. First, the position of the RCM can be computed as:

$$x_{rcm}(q) = x_{ee}(q) + \lambda_l(q)\hat{l}(q) \tag{5.18}$$

where $x_{rcm}(q) \in \mathbb{R}^3$ is the Cartesian position of the RCM, $x_{ee}(q) \in \mathbb{R}^3$ is the Cartesian position of the end-effector, $\hat{l}(q) \in \mathbb{R}^3$ is the unit vector representing the tool direction

(e.g. the z axis of the end-effector) and:

$$\lambda_l(q) = \hat{l}^T(q)(x_{rcm}^{des} - x_{ee}(q)) \tag{5.19}$$

where $x_{rcm}^{des} \in \mathbb{R}^3$ is the desired Cartesian position of the RCM.

Once the end-effector of the robot is endowed with the laparoscopic tool and the tool axis is defined with respect to the end-effecor, Equation (5.18) can be exploited to compute the actual RCM position $x_{rcm}(q)$. Figure 5.2 reports a schematic representation of how the RCM position is computed.

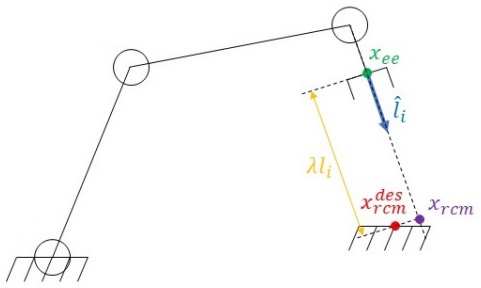


Figure 5.2: Schematic representation of how the RCM position is computed.

I can ba noticed that x_{rcm} as in (5.18) depends only on the configuration q and by the desired position of the RCM x_{rcm}^{des}. This means that, once x_{rcm}^{des} has been defined, differentiating (5.18) with respect to time leads to:

$$\dot{x}_{rcm} = \frac{\partial x_{rcm}}{\partial q}\dot{q} \tag{5.20}$$

Therefore, a mapping between the joint velocities and the RCM velocity can be defined through the Jacobian matrix $J_{rcm} \in \mathbb{R}^{3\times n}$ as:

$$J_{rcm} = \frac{\partial x_{rcm}}{\partial q} \tag{5.21}$$

and the RCM velocity can be computed as:

$$\dot{x}_{rcm} = J_{rcm}\dot{q} \tag{5.22}$$

Once the position and the velocity of the RCM has been computed, a feedback controller for the improvement of the RCM tracking can be designed setting:

$$\tau^{rcm} = J_{rcm}^T\Big(K_{rcm}\big(x_{rcm}^{des} - x_{rcm}\big) - D_{rcm}\dot{x}_{rcm}\Big) \tag{5.23}$$

where $K_{rcm}, D_{rcm} \in \mathbb{R}^{3\times 3}$ are the proportional and derivative gains of the RCM error.

Figure 5.3 reports a schematic representation of the overall controller.

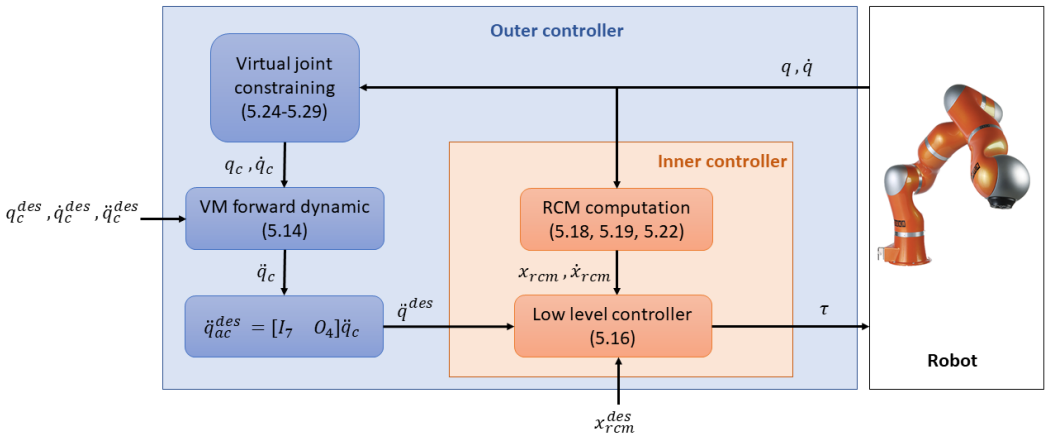


Figure 5.3: Schematic representation of the overall controller.

5.3.3 Virtual Joints Computation

Given the configuration q of RM, the corresponding configuration q_c of VM can be computed by setting:

$$q_c = \begin{bmatrix} q & q_c^{n+1} & \dots & q_c^{n+4} \end{bmatrix}^T \tag{5.24}$$

The first n component of q_c represents the configuration of RM. The first virtual joint q_c^{n+1} consists of a linear guide, with the same axis of the tool and link position coincident with the RCM position x_{rcm}. For this reason, q_c^{n+1} can be computed as in (5.19) as:

$$q_c^{n+1} = \hat{l}^T(q)(x_{rcm}^{des} - x_{ee}(q)) \tag{5.25}$$

where symbols are defined in Section 5.3.2. The remaining virtual joints q_c^{n+2},q_c^{n+3},q_c^{n+4} represents a virtual spherical joint with ending orientation equal to the orientation of the base frame, and can be computed as:

$$q_c^{n+2} = \text{atan2}(R_{2,3}, R_{1,3}) \tag{5.26}$$

$$q_c^{n+3} = \text{atan2}(\sqrt{R_{1,3}^2 + R_{2,3}^2}, R_{3,3}) \tag{5.27}$$

$$q_c^{n+4} = \text{atan2}(R_{3,2}, -R_{3,1}) \tag{5.28}$$

where $R = R(q) \in \mathbb{R}^{3\times3}$ is the rotation matrix that represents the orientation of the end-effector of RM with respect to the base frame, and $R_{i,j} \in \mathbb{R}$ represents the element of R at the i-th rows and j-th column.

Once the configuration of has been computed, the joint velocity of VM can be computed starting from the velocity $\dot{q}$ of RM, computing the end-effector twist and re-computing the closed chain joints velocity $\dot{q}_c$ of VM exploiting the closed chain Jacobian, as described in [134]:

$$\dot{q}_c = J_c(q_c)\dot{x} \tag{5.29}$$

where $J_c(q_c) \in \mathbb{R}^{m\times6}$ is the closed chain Jacobian matrix.

5.4 Validation

This section reports the simulations and the experiments performed in order to test the validity of the proposed method.

This validation aims to show that the user can freely choose a control action to perform a task, independently of the RCM. Then, the user can implement the control action directly to RM (hereafter referred to as the *unconstrained scenario*), or to VM with the proposed framework to perform the same task but complying with the RCM (hereafter referred as the *constrained scenario*).

For this validation, a trajectory tracking task using a computed torque control action is performed.

Both simulations and experiments were performed using a KUKA LWR 4+ 7-DOFs robot with a laparoscopic-like tool mounted at the end-effector, and by planning an RCM compliant helical trajectory for the tool-tip movement in the Cartesian space, as represented by the blue markers in Figure 5.4. The Cartesian trajectory was converted into the joint space and used as the reference for both the *unconstrained scenario* (i.e. RM) and the *constrained scenario* (i.e. VM).

As reported in Section 5.3, the main parameters of the model used for VM in the *outer controller* can be freely chosen by the user. For this reason, consider a locally gravity compensated manipulator and neglect the centrifugal and Coriolis term due to the low speed requested by the planned trajectory. The mass matrix M_c was computed starting from a desired mass matrix in the Cartesian space and then translated into the joint space as:

$$M_c(q_c) = \big(J(q_c)\big)^+ \big(B_c^{des}\big)^{-1} \big(J(q_c)^T\big)^+ \tag{5.30}$$

where $B_c^{des} \in \mathbb{R}^6$ is the desired mass matrix in the Cartesian space and $J(q_c) \in \mathbb{R}^{6\times m}$ is the Jacobian of VM.

For the *constrained scenario*, the computed torque controller can be implemented exploiting the *inner controller* and by setting:

$$\tau_c = M_c(q)\Big(\ddot{q}_c^{\,des} + K_c(q_c^{des} - q_c) + D_c(\dot{q}_c^{\,des} - \dot{q}_c)\Big) \tag{5.31}$$

in the *outer controller*, where $\ddot{q}_c^{\,des}, \dot{q}_c^{\,des}, q_c^{des} \in \mathbb{R}^m$ are the desired acceleration, velocities and positions respectively, and $K_c, D_c \in \mathbb{R}^{n\times n}$ are the controller stiffness and damping respectively.

For the *unconstrained scenario*, the same controller can be implemented by removing both the *outer controller* and the *inner controller* and by directly setting:

$$\tau = M(q)\Big(\ddot{q}^{\,des} + K(q^{des} - q) + D(\dot{q}^{\,des} - \dot{q})\Big) + C(q,\dot{q})\dot{q} + G(q) \tag{5.32}$$

where:

$$q^{des} = \begin{bmatrix} I_n & O_{n\times 4} \end{bmatrix} q_c^{des}$$

and:

$$K = \begin{bmatrix} I_n & O_{n\times 4} \end{bmatrix} K_c \begin{bmatrix} I_n \\ O_{4\times n} \end{bmatrix}, \quad D = \begin{bmatrix} I_n & O_{n\times 4} \end{bmatrix} D_c \begin{bmatrix} I_n \\ O_{4\times n} \end{bmatrix}$$

The stiffness values $K \in \mathbb{R}^n$ and the damping values $D \in \mathbb{R}^n$ were set equal to the first n component of the stiffness values K_c and the damping values D_c used in the *constrained scenario*, to achieve a more truthful comparison.

The trajectory was composed of 500 points with a completion time of $25s$. The same trajectory was linearly interpolated in the joint space to provide to the controller the correct reference. A low number of points and an interpolation on the joint space are used in order to bother both the trajectory tracking and the RCM tracking.

Table 5.1 reports the controller gains used for both simulations and experiments while Table 5.2 reports the proportional and derivative gains used in the RCM tracking improvement term (see Equation (5.23)).

Joint	Simulations		Experiments	
	Stiffness $\left[\frac{Nm}{rad}\right]$	Damping $\left[\frac{Nm\cdot s}{rad}\right]$	Stiffness $\left[\frac{Nm}{rad}\right]$	Damping $\left[\frac{Nm\cdot s}{rad}\right]$
1	1000	100	1000	30
2	1000	100	1000	30
3	1000	100	100	1
4	1000	100	1000	30
5	1000	100	500	10
6	1000	100	300	10
7	1000	100	200	10
8	1000	100	10	1
9	1000	100	10	1
10	1000	100	40	1
11	1000	100	10	1

Table 5.1: Controller joints stiffness and damping parameters used for the simulation and the experiments.

Axis	Simulations		Experiments	
	Stiffness $\left[\frac{Nm}{rad}\right]$	Damping $\left[\frac{Nm\cdot s}{rad}\right]$	Stiffness $\left[\frac{Nm}{rad}\right]$	Damping $\left[\frac{Nm\cdot s}{rad}\right]$
x	100	2	5000	200
y	100	2	5000	200
z	100	2	5000	200

Table 5.2: Controller RCM stiffness and damping parameters used for the simulation and the experiments.

Remark. *In the unconstrained scenario the contribution τ^{rcm} can not be simply added to the controller action* (5.32) *since the control action due to* (5.32) *and the term τ^{rcm} are not compatible with each other, i.e. they are not both RCM compliant. In general, it is not guaranteed that by adding two terms, which individually produce two effects, the sum of the effects is obtained. Simulations and experiments show that this does not hold when adding τ^{rcm} in* (5.16)*. Substituting* (5.16) *into* (5.1) *leads to:*

$$M(q)\ddot{q} = M(q)\ddot{q}^{\,des} + \tau^{rcm} \tag{5.33}$$

and the two terms $M(q)\ddot{q}^{des}$ *and* τ^{rcm} *are both RCM compliant for construction, and then, compatible with each other.*

5.4.1 Simulations

Simulations were initially performed tracking the original 500 points trajectory for both the *unconstrained scenario* and the *constrained scenario*. Then, the number of points of the commanded trajectory was decreased to 250, 125, and 63.

All the simulations are performed using MATLAB® and the Robotic Toolbox [111], with a constant discrete time step of 1ms.

The results of the simulation are reported in Figures 5.4, 5.6, and 5.8 for the *unconstrained scenario* and in Figures 5.5, 5.7, and 5.9 for the *constrained scenario*.

Figure 5.4 and Figure 5.5 report the tool-tip trajectory for the *unconstrained scenario* and the *constrained scenario* respectively. As visible looking at both the figures, the quality of the tracking decreases as the number of points used to describe the desired trajectory decreases, as expected.

Despite this, the worsening introduced by the low number of points is visibly more contained when the trajectory is performed using the proposed controller (i.e. the *constrained scenario*), as visible comparing Figures 5.6 and 5.7, which report the tool-tip trajectories tracking error norms for both the *unconstrained scenario* and the *constrained scenario*.

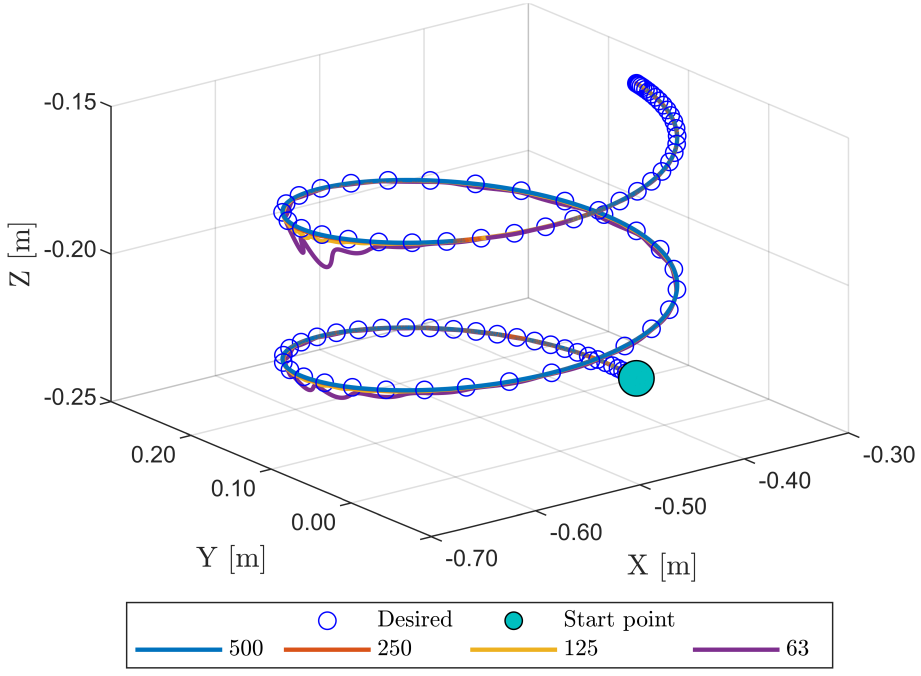


Figure 5.4: Tool-tip trajectories for the *unconstrained scenario*.

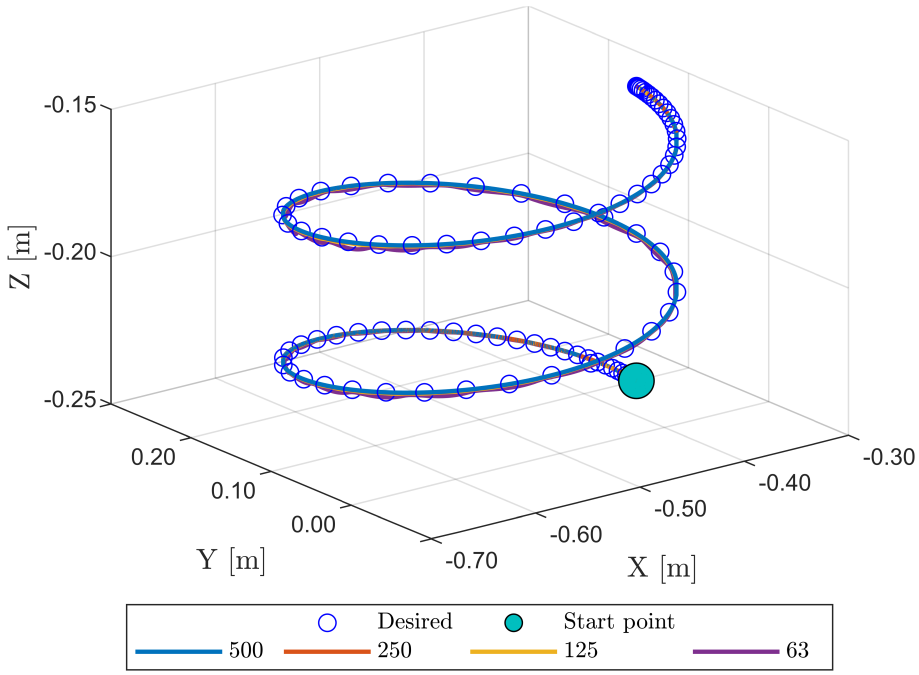


Figure 5.5: Tool-tip trajectories for the *constrained scenario*.

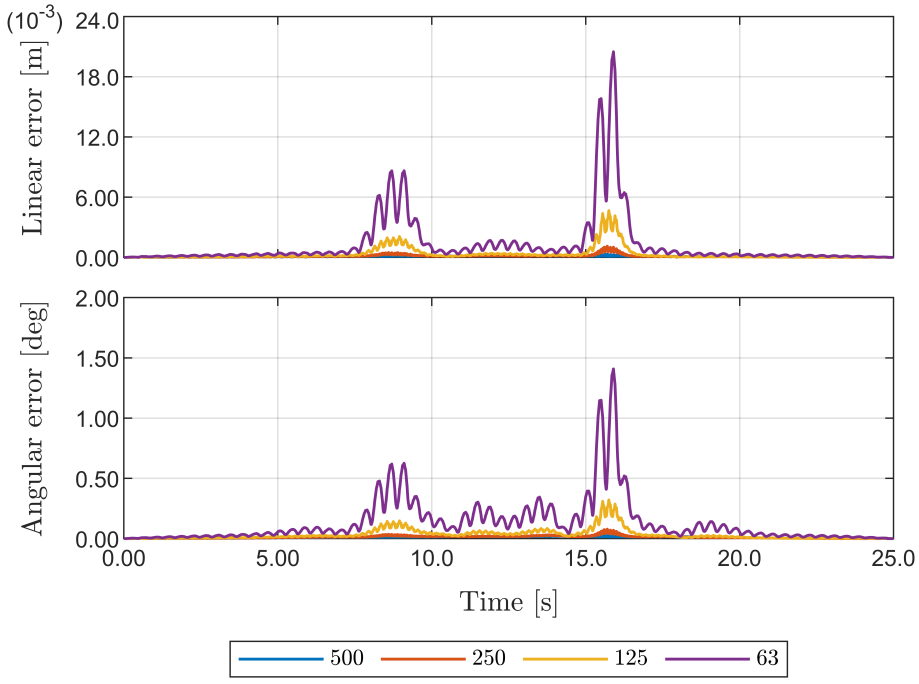


Figure 5.6: Tool-tip trajectories tracking error norms for the *unconstrained scenario*.

Figure 5.8 and Figure 5.9 reports the norm error between the desired RCM position and the RCM real position for the same simulations. The same considerations of the tool-tip hold for the RCM error. Decreasing the number of points used to describe the desired trajectory increases the RCM error, in particular for the *unconstrained scenario*. The error is drastically reduced in the *constrained scenario* where the proposed controller is used, as expected. This highlights the effectiveness of the proposed strategy.

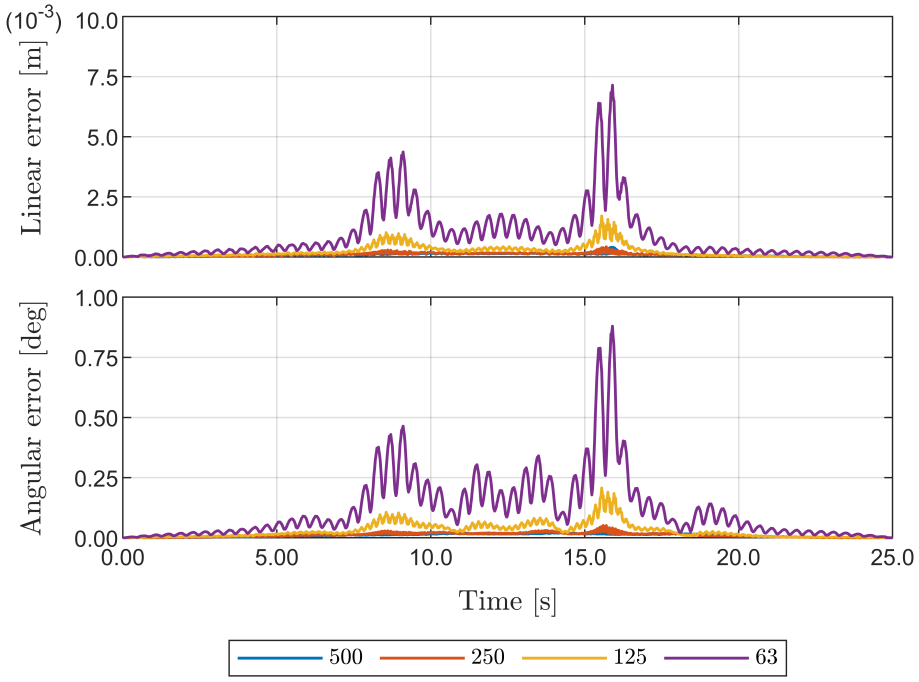


Figure 5.7: Tool-tip trajectories tracking error norms for the *constrained scenario*.

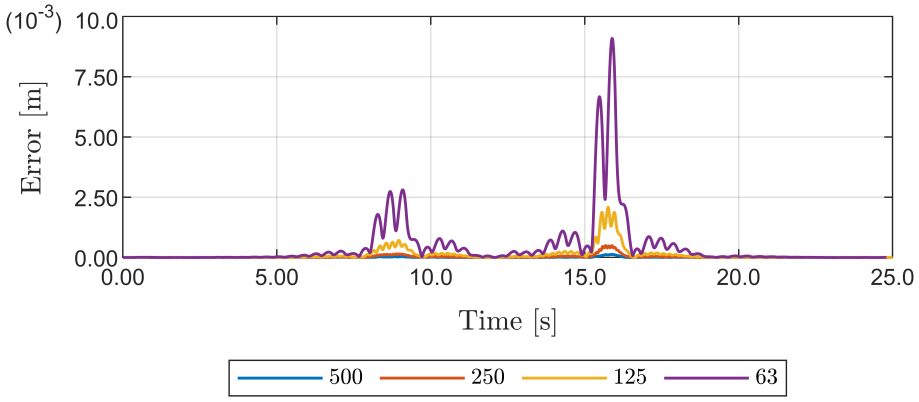


Figure 5.8: RCM error norms for the *unconstrained scenario*.

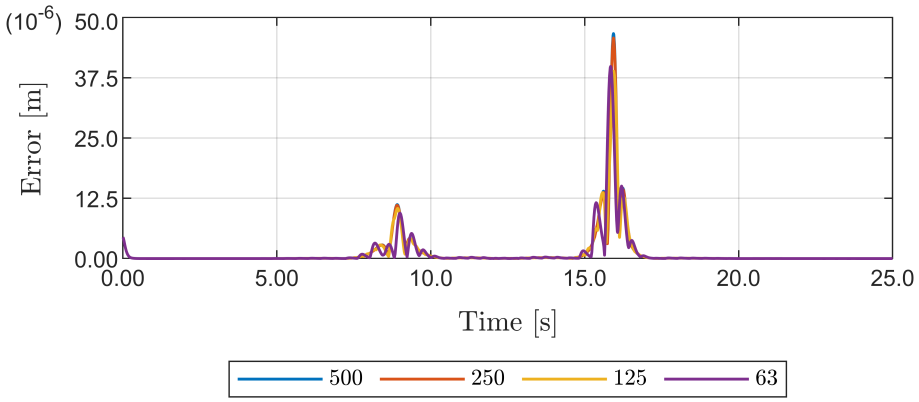


Figure 5.9: RCM error norms for the *constrained scenario*.

In order to prove the effectiveness of the proposed architecture with respect to the state of the art, a comparison with the strict priority method proposed in [126] is performed.

The controller in [126] can be implemented by setting:

$$\tau = \tau_1 + N_2\tau_2 \tag{5.34}$$

where $\tau_1 \in \mathbb{R}^n$ is the torque control vector of the primary task, $N_2 \in \mathbb{R}^{n\times n}$ is the null-space projector of the secondary task and $\tau_2 \in \mathbb{R}^n$ is the torque control vector of the secondary task.

The term τ_1 is defined as:

$$\tau_1 = \bar{J}_{rcm}^T\big(-K_1\zeta - D_1\dot{\zeta}\big) \tag{5.35}$$

where $\zeta \in \mathbb{R}$ describes the task and it is defined as:

$$\zeta = -D = -\hat{D}^T(x_{rcm}^{des} - x_{rcm}) \tag{5.36}$$

with $\hat{D}$ representing the unit vector in the direction from x_{rcm} toward x_{rcm}^{des}.

The term $\bar{J}_{rcm} \in \mathbb{R}^{1\times n}$ can be computed as:

$$\bar{J}_{rcm} = \hat{D}^T J_{rcm} \tag{5.37}$$

By setting τ_1 as reported above, the primary task takes care of maintaining the RCM constraint. As a secondary task, the computed torque controller is projected by setting:

$$N_2 = I - \bar{J}_{rcm}^T\big(\bar{J}_{rcm}^+\big)^T \tag{5.38}$$

and:

$$\tau_2 = M(q)\Big(\ddot{q}^{\,des} + K(q^{des} - q) + D(\dot{q}^{\,des} - \dot{q})\Big) + C(q,\dot{q})\dot{q} + G(q) \tag{5.39}$$

Table 5.3 reports the maximum error norm between the RCM desired position and the RCM real position for all the simulations conducted. In both the *unconstrained scenario* and with the controller proposed in [126], the maximum RCM error increase easily, exceeding safe values. This does not hold for the *constrained scenario*, where the maximum RCM error remains constant and below a safe value for all the simulations, emphasizing the effectiveness of the proposed method to correctly maintain the RCM constraint while performing a task. Moreover, the steadiness with which the RCM error is maintained varying the trajectory points number underlines the robustness of the proposed method with respect to [126].

Scenario	Trajectory points				Units
	500	250	125	63	
unconstrained	0.1306	0.5002	2.0889	9.0893	[mm]
constrained	0.0467	0.0457	0.0389	0.0398	[mm]
[126]	0.0324	0.4125	0.6761	1.5550	[mm]

Table 5.3: Comparison of the maximum RCM error norms varying the number of points with which the trajectory is planned.

Trajectories performed with the controller proposed in [126] are not reported since they are very similar to those shown in Figure 5.5.

5.4.2 Experiments

Two different experiments are performed. The first experiment replicates the simulations, using the 500 points planned trajectory, to validate the practical effectiveness of the proposed controller. For the second experiment, two different RCM compliant configurations are chosen. Hence, a point-to-point trajectory that moves the robot from one configuration to the other was computed using a 5-th order polynomial for each joint. This trajectory was then used as the reference for both the *unconstrained scenario* and the *constrained scenario*. This trajectory clearly violates the position of the RCM. Indeed, this second experiment aims to validate the behavior of the proposed controller in extreme conditions, proving its real flexibility. Both the experiments were performed with a controller step time of 1ms.

The results of the first experiment are depicted in Figures 5.10, 5.11, and 5.12, which reports the tool-tip trajectory, the tool-tip trajectory tracking error norm, and the RCM error norm, respectively. Results are reported for only the *constrained scenario* since the comparison between the two scenarios has been largely covered in the previous subsection.

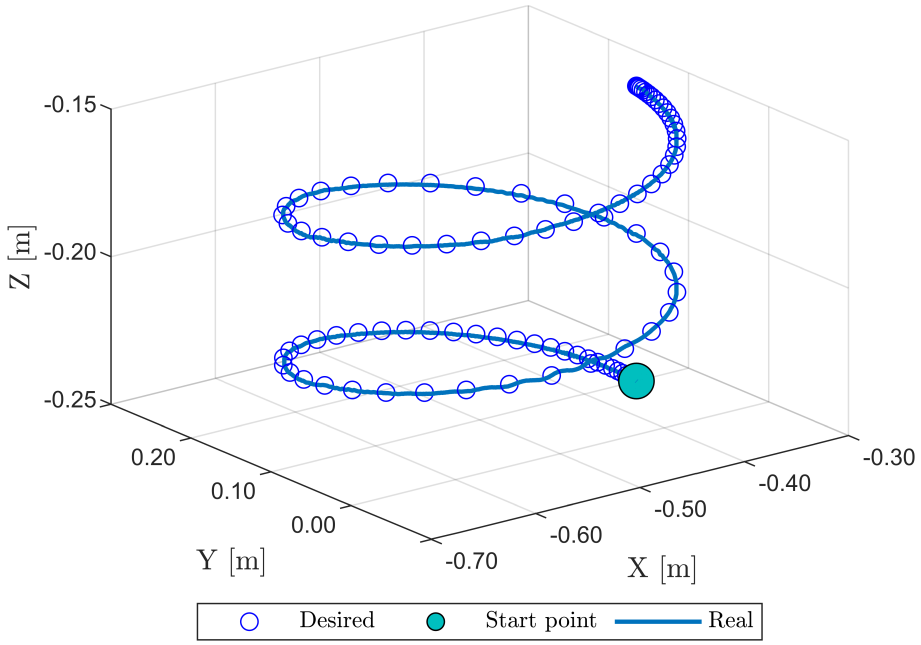


Figure 5.10: Tool-tip trajectory in the first experiment.

As expected, the robot tracks the desired trajectory with good tracking performance, from both the trajectory and the RCM point of view. The controller allows to keep the trajectory tracking error near the millimeter and to increase the RCM position accuracy from $2.33mm$ to $0.79mm$, with respect to the same trajectory tracked in the *unconstrained scenario*. The relevant improvement of the RCM position accuracy allows to state the effectiveness of the proposed controller also for real experiments.

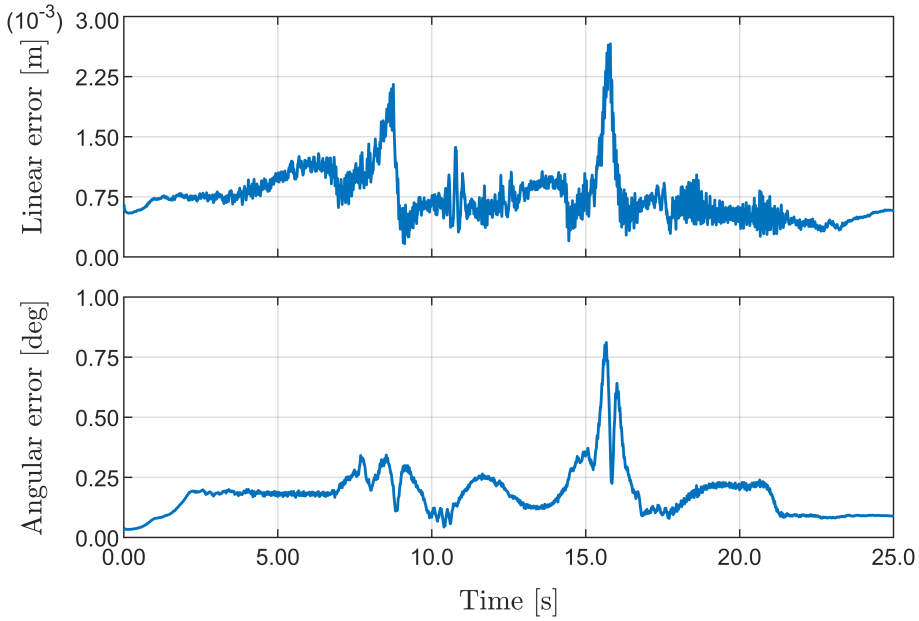


Figure 5.11: Tool-tip trajectory tracking error norm in the first experiment.

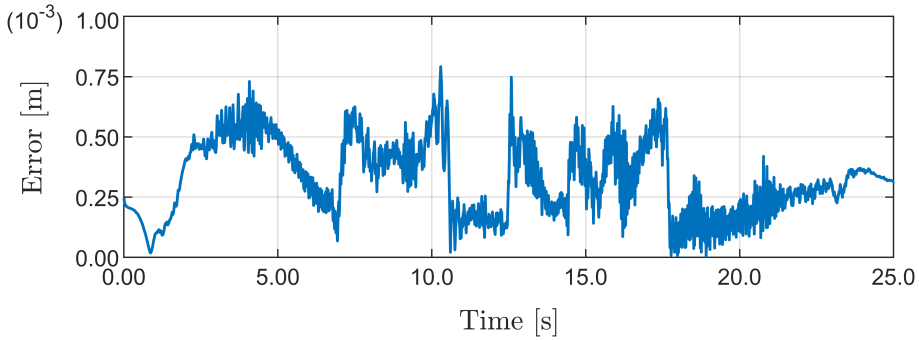


Figure 5.12: RCM error norm in the first experiment.

The RCM error was computed using Equation (5.18) with the direct kinematic performed using the measurements of the robot encoders.

The results of the second experiment are reported in Figures 5.13, 5.14, and 5.15.

As evident from Figure 5.13, the two controllers move the robot following different trajectories. In the *unconstrained scenario*, the commanded trajectory was tracked with high fidelity but reporting an expected unacceptable high maximum RCM error norm of 19.60mm. In the *constrained scenario* the original trajectory was modified by the controller degrading the trajectory tracking performance but allowing to register a maximum RCM error norm of 0.69mm.

The differences between the commanded trajectory and the performed one, accompanied by the significant decrease in the registered maximum RCM error norm allows to state that the controller is able to change the behavior of the robot such that the RCM constraint is always guaranteed.

All the results allow to state the effectiveness of the proposed strategy.

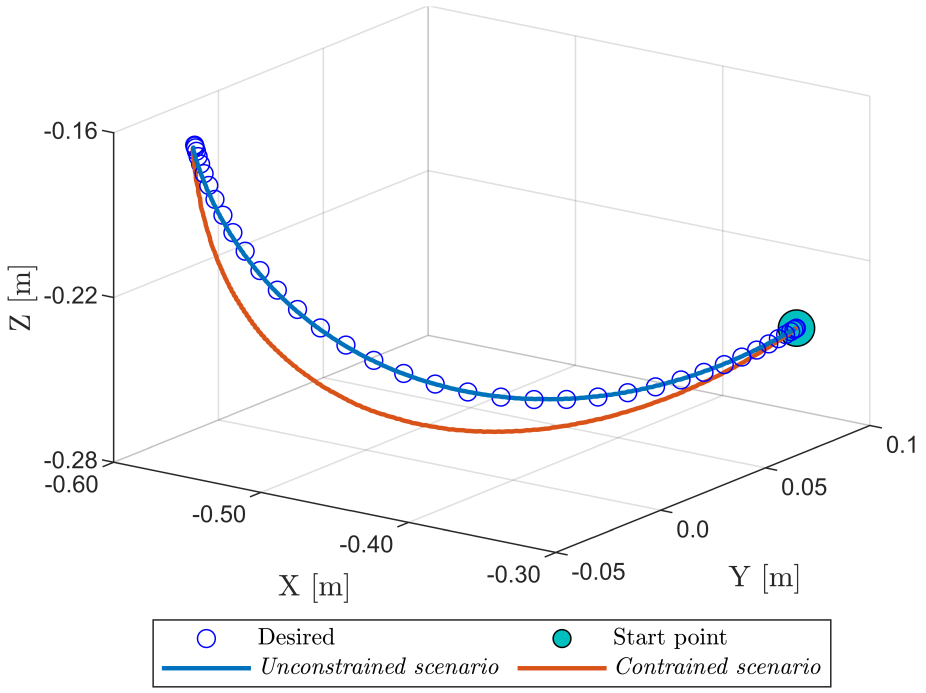


Figure 5.13: Tool-tip trajectory comparison between the *unconstrained scenario* and the *constrained scenario* in the second experiment.

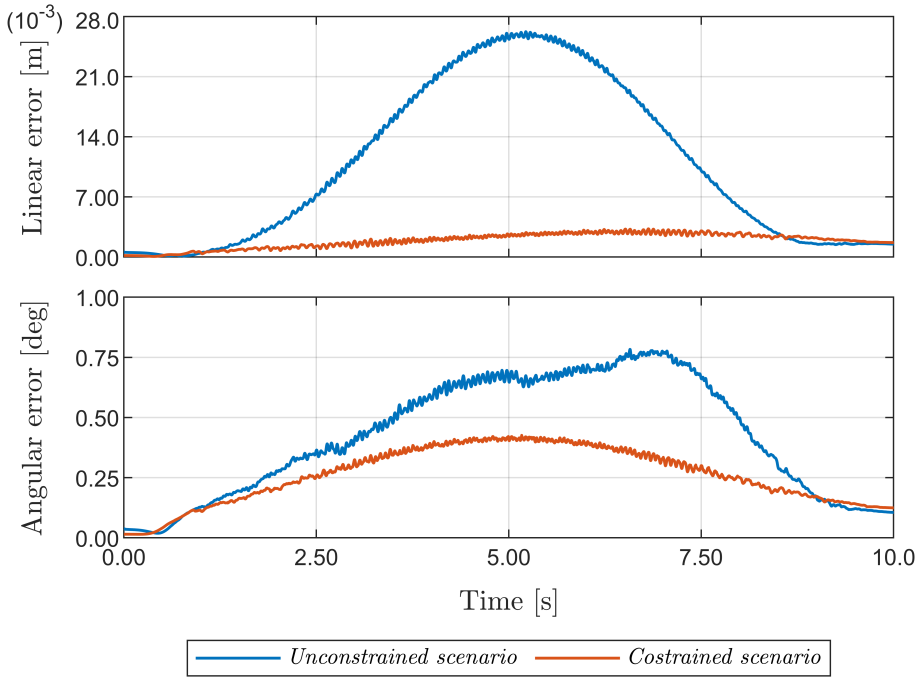


Figure 5.14: Tool-tip trajectory tracking error norms comparison between the *unconstrained scenario* and the *constrained scenario* in the second experiment.

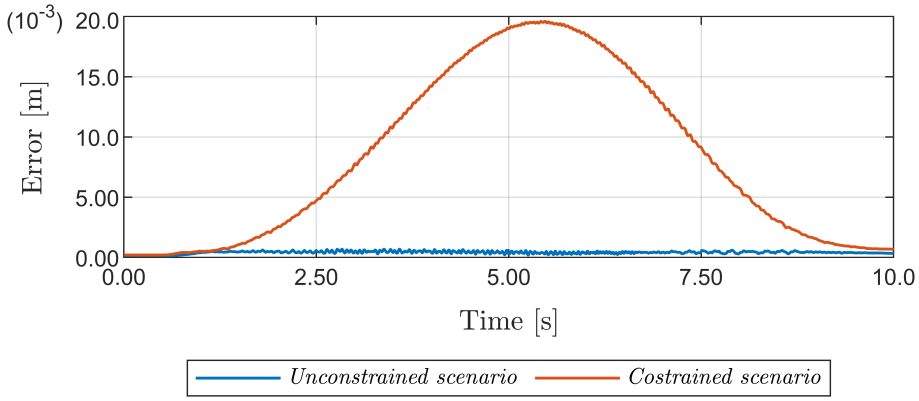


Figure 5.15: RCM error norms between the *unconstrained scenario* and the *constrained scenario* in the second experiment.

5.5 RCM Controller V2.0

The torque controller presented so far was developed by formulating the RCM constraint exploiting closed chain manipulator theory and by using virtual joints. The major issue of this controller is that the system emulates the dynamic of a virtual system, making difficult to find a relationship between the emulated dynamics and the real one. This did not allow the system to effectively interact with the environment and, therefore, to implement a compliant motion control, e.g. impedance control.

To solve this problem and improve the capabilities of the system, the dynamic constraint method used in Section 5.3.1 is used to directly constrain the dynamics of the manipulator (5.1), allowing to remove the simulation of virtual joints and decrease the computational load. Moreover, this allows to implement more effective controllers.

5.5.1 Controller

Constraining Method

To synthesize the controller, the same constraining method used in Section 5.3.1 is used to constraint (5.1). This method can be summarized as in the following:

- consider a generic m-DOFs manipulator represented by the Euler-Lagrange model (5.2).
- consider that the system in (5.2) is subject to a set of kinematic constraints modeled as (5.8).
- the dynamic model of the manipulator (5.2) subject to the constraint (5.8) is represented by (5.9) with (5.10), (5.11) and (5.12).

RCM Constraint

Since the RCM constraint reported in Section 5.3.1 was developed for the virtual manipulator VM and not for the real manipulator RM, a new description of the RCM constraint must be formulated for (5.1). This can be done by setting:

$$x_{rcm}(q) = x_{rcm}^{des} \tag{5.40}$$

where $x_{rcm}(q) \in \mathbb{R}^3$ and $x_{rcm}^{des} \in \mathbb{R}^3$ are the real and the desired Cartesian position of the RCM, respectively, and $x_{rcm}(q)$ can be computed using (5.18) and (5.19).

The term x_{rcm}, already defined in (5.18), depends only on the configuration q and by the desired position of the RCM x_{rcm}^{des}. This means that, once x_{rcm}^{des} has been defined, differentiating (5.40) with respect to time twice leads to:

$$J_{rcm}\ddot{q} + \dot{J}_{rcm}\dot{q} = 0 \tag{5.41}$$

Equation (5.41) can be rewritten as:

$$J_{rcm}\ddot{q} = -\dot{J}_{rcm}\dot{q} \tag{5.42}$$

that has the same form as (5.8). Indeed, setting:

$$A_{rcm} = J_{rcm}, \quad b_{rcm} = -\dot{J}_{rcm}\dot{q} \tag{5.43}$$

lead to:

$$A_{rcm}\ddot{q} = b_{rcm} \tag{5.44}$$

This means that using (5.44) and the constraining strategy reported in the previous section on (5.1) allows to compute the dynamic model of (5.1) subject to the RCM constraint.

Controller Synthesis

As already mentioned in Section 3.3, the control torque τ for (5.1) have to design such that the motion of the system is always compliant with the RCM, leaving the user to freely design the desired behavior.

In order to guarantee the RCM constraint, the effects of the external force need to be compensated by the controller. In this way, an interaction with the environment will never produce an RCM non-compliant behavior. This can be simply achieved by setting:

$$\tau = \bar{\tau} - \tau^{ext} \tag{5.45}$$

where $\bar{\tau} \in \mathbb{R}^n$ is the new control input vector.

Substituting (5.45) in (5.1) it follows that:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \bar{\tau} \tag{5.46}$$

in which the external force effect has been compensated.

The model in (5.46) has the same form as the model in (5.2). Following the approach reported in Section 5.3.1 and summarized in Section 5.5.1, (5.46) can be constrained using (5.44) by setting:

$$\bar{\tau} = Q(q,\dot{q}) + Q^{constr}(q,\dot{q}) + C(q,\dot{q})\dot{q} + G(q) \tag{5.47}$$

where:

$$Q(q,\dot{q}) = \tau^{\star} - C(q,\dot{q})\dot{q} - G(q) \tag{5.48}$$

$$Q^{constr}(q,\dot{q}) = M(q)^{\frac{1}{2}} B^{+}(q)\Big(b_{rcm}(q,\dot{q}) - A_{rcm}(q,\dot{q})M(q)^{-1}Q(q,\dot{q})\Big) \tag{5.49}$$

$$B(q) = A_{rcm}(q,\dot{q})M(q)^{\frac{1}{2}} \tag{5.50}$$

and $\tau^{\star}$ is the new control input.

Substituting (5.47) in (5.46) leads to:

$$M(q)\ddot{q} = Q(q,\dot{q}) + Q^{constr}(q,\dot{q}) \tag{5.51}$$

that, with the same form of (5.9), is equivalent to the dynamic model of the manipulator (5.46) subject to the RCM constraint (5.40). This means that setting $\bar{\tau}$ as in (5.47) will make the system behave compliantly with the RCM, regardless of the choice of $\tau^{\star}$. This means that the desired control action term $\tau^{\star}$ can be freely chosen by the user disregarding the RCM constraint. The term $\tau^{\star}$ can be then employed in (5.47) to achieve the RCM compliant version of the desired control action. This confers extreme flexibility to the system.

RCM Enforcement

As mentioned in Section 5.3.2 the RCM constraint (5.44) depends on the positions q and velocities $\dot{q}$ and works on the acceleration level $\ddot{q}$. This means that misalignment in terms of the RCM position will occur when the integration time of the model is finite (e.g. discrete-time controller) and/or when the model of the system is not perfectly known. These misalignments cause errors in evaluating the constraint function (5.44) and then in the evaluation of constraining action (5.49), rapidly degrading the overall performance.

In order to enforce the tracking performance of the RCM position, the same term τ^{rcm} as (5.23) in Section 5.3.2 is added to the overall control action (5.47). This term implements a feedback controller and monitors the position and the speed of the RCM during the motion of the system. This enforces the tracking of the RCM position, improving the evaluation of the constraint (5.44) and of the constraint torque (5.49). This allows to increase the performance of the overall system.

Overall Controller

The overall controller for the system in (5.1) can be synthesized considering (5.47), (5.45), and (5.23) as:

$$\begin{aligned}\tau = Q(q,\dot{q}) + Q_{constr}(q,\dot{q}) + C(q,\dot{q})\dot{q} + G(q) - \tau^{ext} + \\ + J_{rcm}^T\Big(K_{rcm}\big(x_{rcm}^{des} - x_{rcm}\big) - D_{rcm}\dot{x}_{rcm}\Big)\end{aligned} \tag{5.52}$$

that makes the system in (5.1) behaves like:

$$M(q)\ddot{q} = Q(q,\dot{q}) + Q_{constr}(q,\dot{q}) + J_{rcm}^T\Big(K_{rcm}\big(x_{rcm}^{des} - x_{rcm}\big) - D_{rcm}\dot{x}_{rcm}\Big) \tag{5.53}$$

with $\tau^\star$ in $Q(q,\dot{q})$ as control input.

Exploiting $\tau^\star$ the user can implement the desired control action. Consider a desired behavior for the manipulator (e.g. impedance control) and let $u \in \mathbb{R}^n$ be the corresponding control input. There are two main options for the implementation:

1. Set $\bar{\tau} = u$ in (5.45), disregarding the RCM constraint.

2. Set $\tau^\star = u$ in (5.52), in order to explicitly consider the RCM constraint.

As for the previous controller, the scenario in which the desired control action is implemented using $\bar{\tau}$ in (5.45) will be hereafter referred to as the *unconstrained scenario*, and scenario in which the desired control action is implemented using $\tau^\star$ in (5.52) as the *constrained scenario*.

5.6 Validation

5.6.1 Simulations

Several simulations were conducted before moving to the implementation on a real manipulator, with the aim of testing the effectiveness of the proposed controller.

As for the previous controller, all the simulations were performed considering a KUKA LWR 4+ 7-DOF robot with a laparoscopic-like tool mounted at the end-effector, and by planning a Cartesian RCM compliant helical trajectory with a completion time of $25s$. The simulations are performed using MATLAB® and the Robotic Toolbox [111], with a constant discrete-time step of 1ms.

All simulations are performed setting:

$$\begin{aligned}K_{rcm} &= \mathrm{diag}(10.0, 10.0, 10.0) \\ D_{rcm} &= \mathrm{diag}(1.0, 1.0, 1.0)\end{aligned} \tag{5.54}$$

Trajectory Following

The first simulation set is a comparison with the previous controller and aims at demonstrating the effectiveness of the proposed controller in performing a trajectory following task while guaranteeing the RCM constraint.

In these simulations, the desired control action is the computed torque control action. This was achieved by setting:

$$\bar{\tau} = C(\dot{q}, q)\dot{q} + G(q) + M(q)\Big(\ddot{q}^{\,des} + K(q^{\,des} - q) + D(\dot{q}^{\,des} - \dot{q})\Big) \tag{5.55}$$

for the *unconstrained scenario*, and:

$$\tau^{\star} = C(\dot{q}, q)\dot{q} + G(q) + M(q)\Big(\ddot{q}^{\,des} + K(q^{\,des} - q) + D(\dot{q}^{\,des} - \dot{q})\Big) \tag{5.56}$$

for the *constrained scenario*, where $\ddot{q}^{\,des}, \dot{q}^{\,des}, q^{\,des}$ are the desired acceleration, velocities and positions respectively, and $K, D \in \mathbb{R}^{n \times n}$ are the controller stiffness and damping respectively.

As for the previous controller, the problem of tracking a helical trajectory represented by a variable number of points, from 500 to 63 is considered.

In order to achieve a more truthful comparison, the parameters K and D are set as in the previous controller (see Table 5.1) as:

$$\begin{aligned} K &= \text{diag}(1000, 1000, 1000, 1000, 1000, 1000, 1000) \\ D &= \text{diag}(100, 100, 100, 100, 100, 100, 100) \end{aligned} \tag{5.57}$$

Simulations performed for the *unconstrained scenario* lead back exactly to those reported in Section 5.4.1. For this reason, they are not reported in this section.

Figures 5.16 and 5.17 reports the trajectory and the trajectory tracking error norms achieved in the *constrained* case, respectively. Both results allows to state that the controller is able to correctly track the desired trajectory independently on the number of points used to describe the trajectory. Figure 5.18 reports the RCM error norm.

Table 5.4 reports the maximum RCM error norm for all the simulations. In the *unconstrained scenario*, the error is higher with respect to the other scenarios and grows when the point used to describe the trajectory decrease. This is an expected behavior as the controller used in this scenario does not take into account the RCM constraint.

Scenario	Trajectory points				Units
	500	250	125	63	
unconstrained	0.1306	0.5002	2.0889	9.0893	[mm]
constrained	0.0470	0.0470	0.0467	0.0470	[mm]
previous controller	0.0467	0.0457	0.0389	0.0398	[mm]

Table 5.4: Comparison of the maximum RCM error norms varying the number of points with which the trajectory is planned.

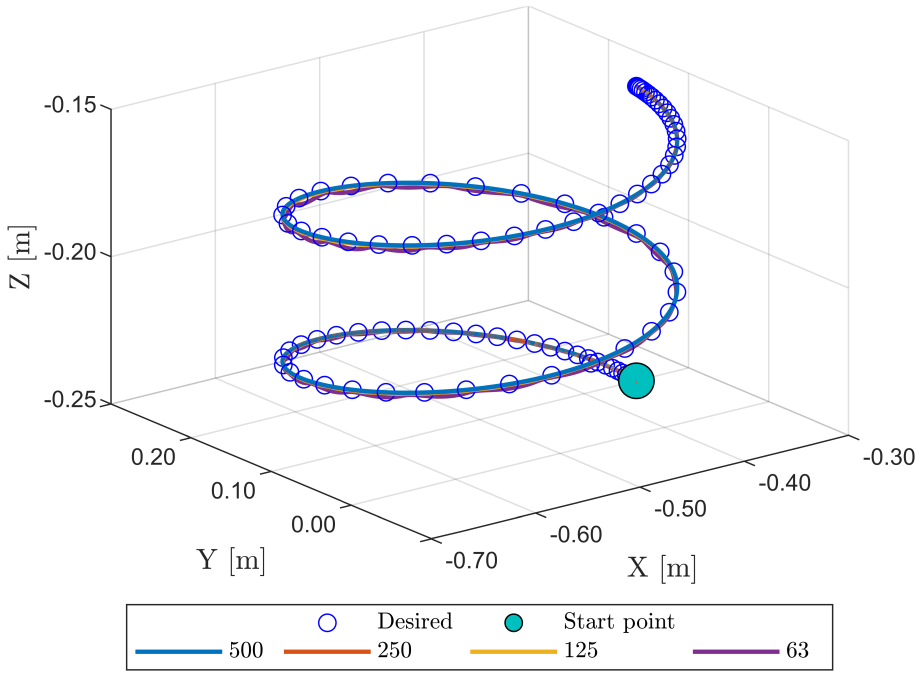


Figure 5.16: Trajectory following - Tool-tip trajectories for the *constrained scenario*.

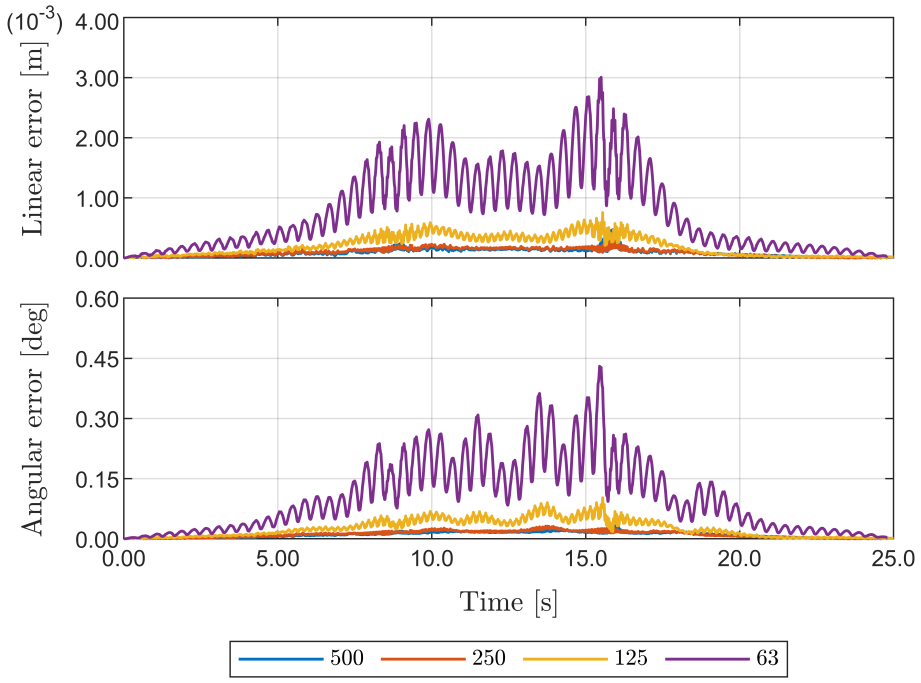


Figure 5.17: Trajectory following - Tool-tip trajectories tracking error norms for the *constrained scenario*.

This does not happen when switching to the *unconstrained scenario*, where the proposed controller is employed. In fact, explicitly considering the RCM constraint allows to effectively maintain the RCM position regardless of the number of points with which the trajectory is described. This allows to state the effectiveness of the proposed controller, which reports a maximum RCM error norm an order of magnitude below the millimeter for all the simulations.

Comparable performances are reported with respect to the previous controller. However, the previous controller does not allow the system to interact effectively with the envi-

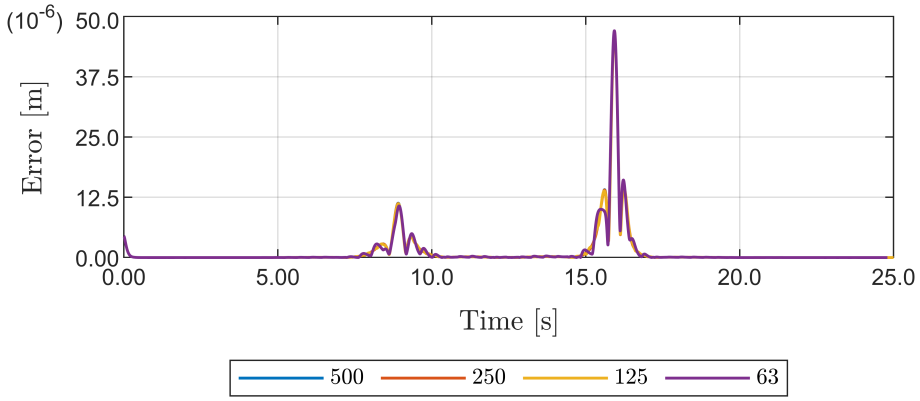


Figure 5.18: Trajectory following - RCM error norms for the *constrained scenario*.

ronment, as it bases its behavior on an emulated dynamics. This does not hold for the proposed controller as it will be shown in the next sections.

Impedance Control

The second simulation aims at demonstrating that the proposed controller can be employed with a compliant motion control and effectively interact with the environment. Here, the desired control action is the impedance control, and is implemented by setting:

$$\begin{aligned} \bar{\tau} = C(q,\dot{q})\dot{q} + G(q) + J(q)^T\Lambda(q)\Big(\ddot{q}^{\,des} + \\ -\dot{J}(q)\dot{q} - (\Lambda^{des})^{-1}(\Pi^{\,des}\dot{e} + \Gamma^{\,des}e + F^{ext})\Big) \end{aligned} \tag{5.58}$$

for the *unconstrained scenario*, and:

$$\begin{aligned} \tau^{\star} = C(q,\dot{q})\dot{q} + G(q) + J(q)^T\Lambda(q)\Big(\ddot{q}^{\,des} + \\ -\dot{J}(q)\dot{q} - (\Lambda^{des})^{-1}(\Pi^{\,des}\dot{e} + \Gamma^{\,des}e + F^{ext})\Big) \end{aligned} \tag{5.59}$$

for the *constrained scenario*, where $\Lambda(q)$, $\Lambda^{des} \in \mathbb{R}^{6\times6}$ are the Cartesian inertia matrix and the desired Cartesian inertia matrix respectively, $J(q) \in \mathbb{R}^{6\times n}$ is the Jacobian matrix and Π^{des}, $\Gamma^{des} \in \mathbb{R}^{6\times6}$ are the controller desired Cartesian stiffness and damping respectively. The term $e = x - x^{des} \in \mathbb{R}^6$ is the Cartesian error between the actual pose $x \in \mathbb{R}^6$ and the desired one $x^{des} \in \mathbb{R}^6$ while $F^{ext} \in \mathbb{R}^6$ is the external Cartesian wrench due to the interaction with the environment.

Substituting (5.58) in (5.45) and then in (5.1), is easy to show that the controller in (5.58) makes the system in (5.1) behaves like a mass-spring-damper system:

$$\Lambda^{des}\ddot{x} + \Pi^{\,des}\dot{x} + \Gamma^{\,des}x = F^{ext} \tag{5.60}$$

In order to simulate the interaction with the environment, a virtual external force F^{ext} in (5.59) of $60N$ is introduced along the y-axis, acting between $t = 10s$ and $t = 20s$.

For this simulation the following parameters were set:

$$\begin{aligned} \Lambda^{des} &= \text{diag}(1.0, 1.0, 1.0, 1.0, 1.0, 1.0) \\ \Pi^{des} &= \text{diag}(100, 100, 100, 10, 10, 10, 100) \\ \Gamma^{des} &= \text{diag}(5000, 5000, 5000, 500, 500, 500) \end{aligned} \tag{5.61}$$

Figures 5.19 and 5.19 reports the tool-tip trajectory and tool-tip trajectory tracking error norms, respectively. As observable, the trajectory is faithfully reproduced in the first part of the simulation, where no external factors affect the execution of the task. Afterward, the action of the external force deviates the behavior of the system in a sensitive way. This behavior has to be attributed to the impedance control implemented, capable of interacting in a compliant way with the external force. In the final section, the trajectory returns to the original one and stops at the final point.

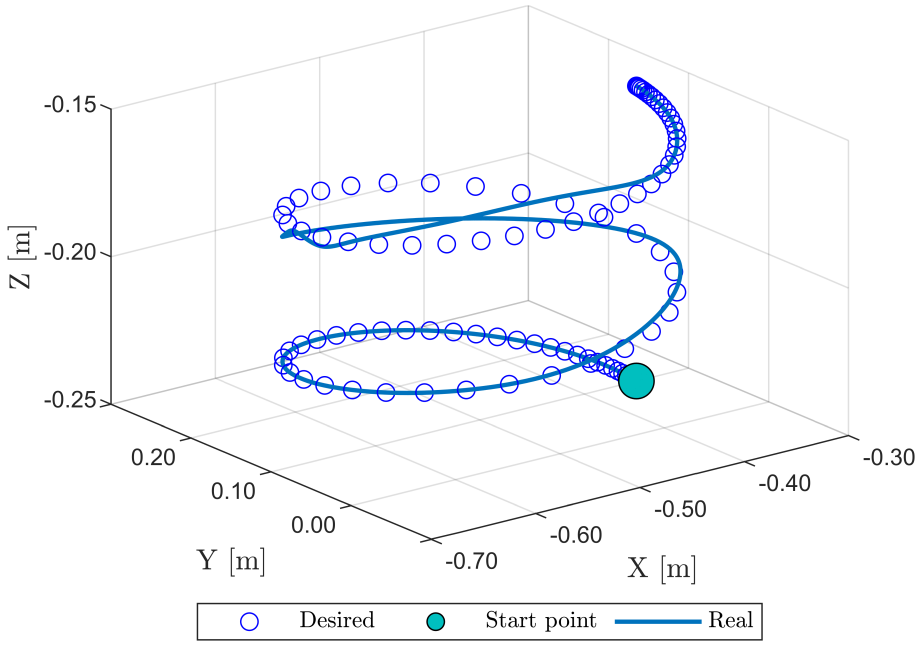


Figure 5.19: Impedance control - Tool-tip trajectory.

Figure 5.21 reports the evolution of the RCM error absolute value along the three axes during the simulation. The error value always remains below $0.01mm$ along all three axes, and has a maximum error norm of $0.009mm$ at $t = 15.736s$, clearly visible by the peak in Figure 5.21. Indeed, in that instant is required a high angular velocity for one of the joints, which leads the system to make a higher error. In any case, the error remains very low.

The correct tracking of the trajectory, the compliance with respect to external forces, and the low RCM error during the execution of the task allow to state that the proposed controller is able to implement compliant motion control actions and interact with the environment, always guaranteeing the RCM constraint satisfaction.

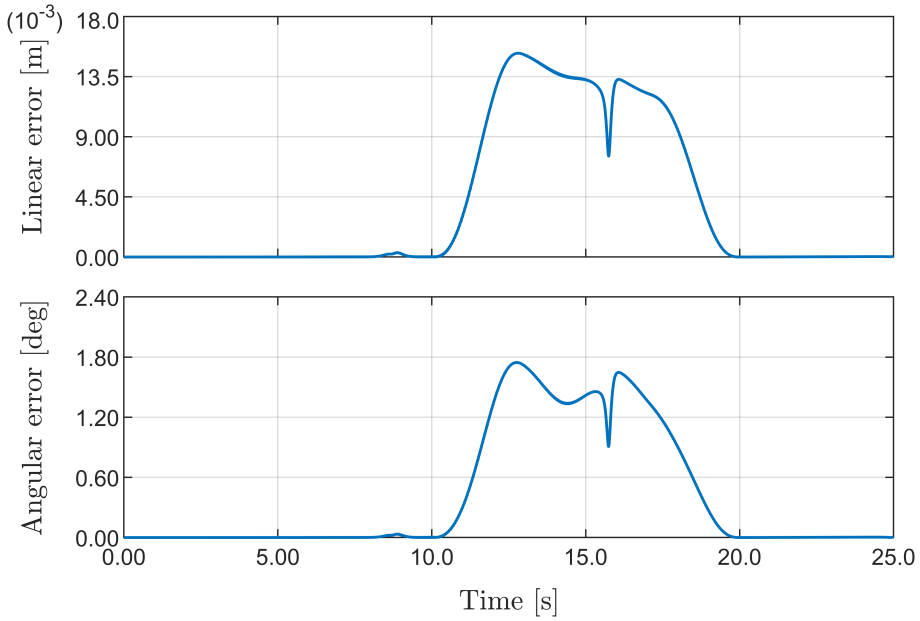


Figure 5.20: Impedance control - Tool-tip trajectory tacking error norms.

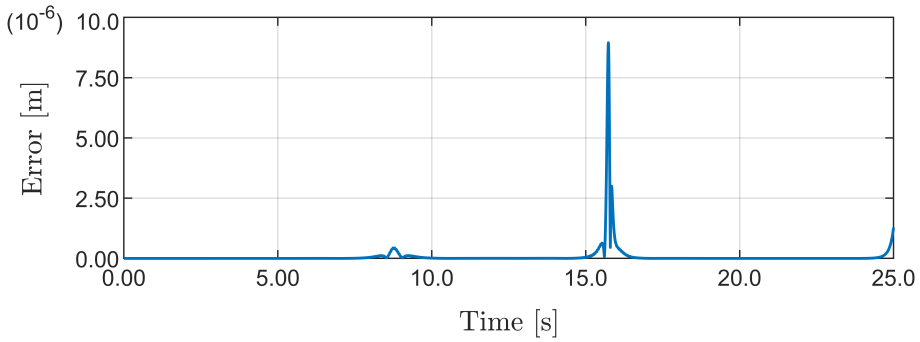


Figure 5.21: Impedance control - RCM error norm.

Comparison with the State of the Art

This simulation reports a comparison between the simulation reported in Subsection 5.6.1 and the same simulation performed using the strict priority method proposed in [126] (see Section 5.4.1) by setting:

$$\tau_1 = \bar{J}_{rcm}^T\big(- K_1\zeta - D_1\dot{\zeta}\big) \tag{5.62}$$

and:

$$\tau_2 = C(q,\dot{q})\dot{q}+G(q)+J(q)^T\Lambda(q)\Big(\ddot{q}^{\,des}-\dot{J}(q)\dot{q}-(\Lambda^{des})^{-1}(\Pi^{\,des}\dot{e}+\Gamma^{\,des}e+F^{ext})\Big) \tag{5.63}$$

with $K_1 = 10.0$, $D_1 = 1.0$ and $\Lambda^{des}, \Pi^{\,des}$ and $\Gamma^{\,des}$ as in (5.61), to have a more truthful comparison.

Figures 5.22, 5.23, and 5.24 reports the results in terms of performed trajectory, trajectory tracking error, and RCM error, respectively. As visible from Figure 5.22 the controller correctly perform the trajectory when no external force is applied (i.e. for $0s < t < 10s$ and $t > 20s$) and, thanks to the impedance control action, is compliant when the

external force acts (i.e. for $10s < t < 20s$). However, this happens with a non-negligible RCM error as visible in Figure 5.24 at $10s < t < 20s$, where a maximum RCM error of $24.2156mm$ is registered. As for the simulation reported in Section 5.6.1 with the proposed controller, the peak error is registered near $t = 15.7$, where a high angular velocity for one of the joints is required. In this scenario, considering the dynamic effect the constraint produced on the dynamic the overall system as our controller does, allows to clearly improve the performance.

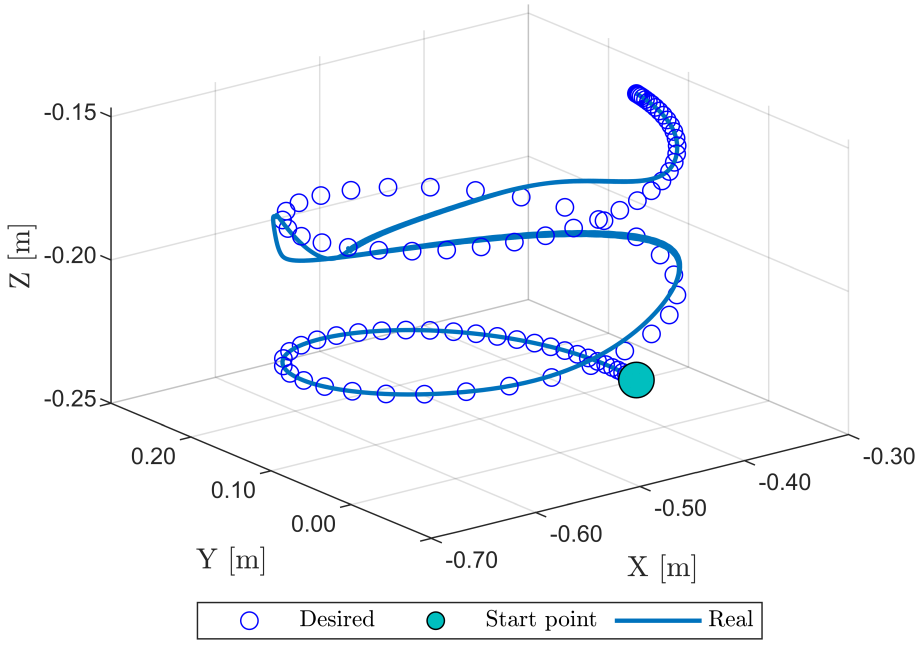


Figure 5.22: Comparison with the state of the art - Tool-tip trajectory.

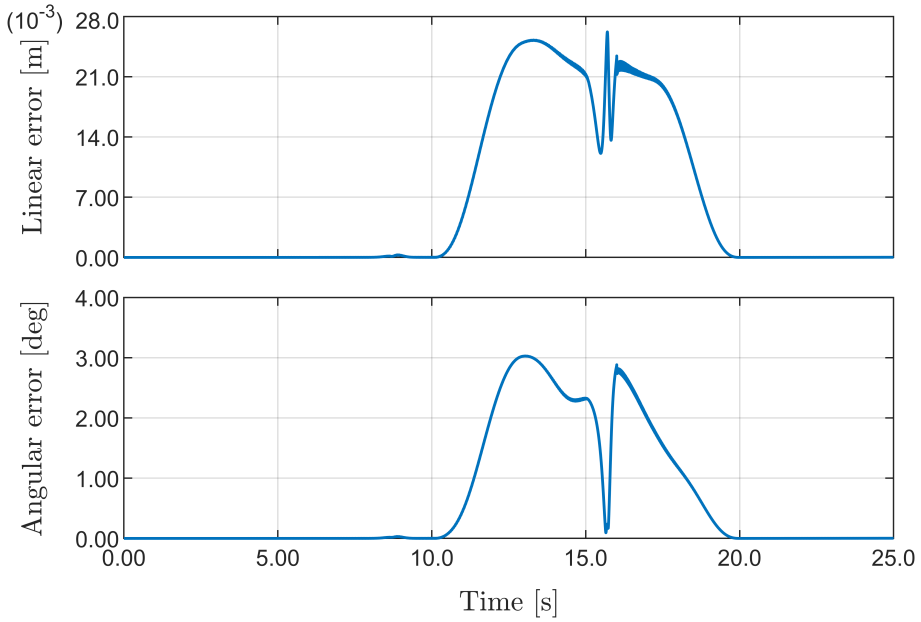


Figure 5.23: Comparison with the state of the art - Tool-tip trajectory tracking error norms.

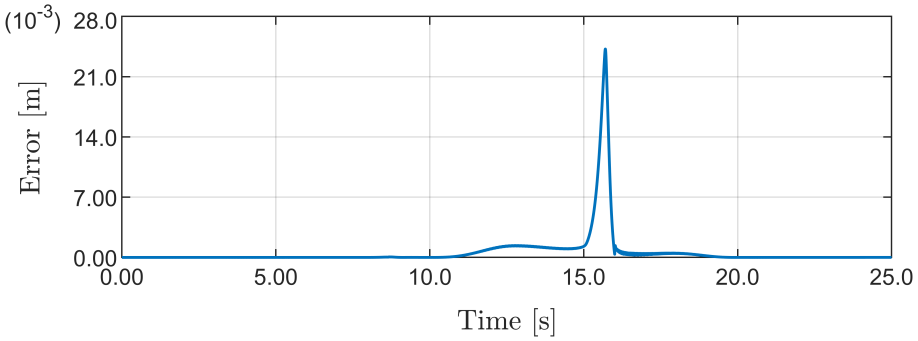


Figure 5.24: Comparison with the state of the art - RCM error norm.

5.6.2 Experiments

In order to prove the effectiveness of the proposed controller also for real manipulators, the second simulation set is replicated experimentally, using the same KUKA LWR 4+ 7-DOF robot with a laparoscopic-like tool mounted at the end-effector. In this case the external force was directly measured using the torque sensors of the robot, and it is introduced by the user interacting with the robot at the end of the trajectory, where the robot track a fixed pose.

For the real experiments the following parameters were set:

$$\begin{aligned} \Lambda^{des} &= \text{diag}(2.5, 2.5, 2.5, 0.5, 0.5, 0.5) \\ \Pi^{des} &= \text{diag}(100, 100, 100, 50, 50, 50) \\ \Gamma^{des} &= \text{diag}(2000, 2000, 2000, 1000, 1000, 1000) \\ K_{rcm} &= \text{diag}(5000, 5000, 5000) \\ D_{rcm} &= \text{diag}(100, 100, 100) \end{aligned} \tag{5.64}$$

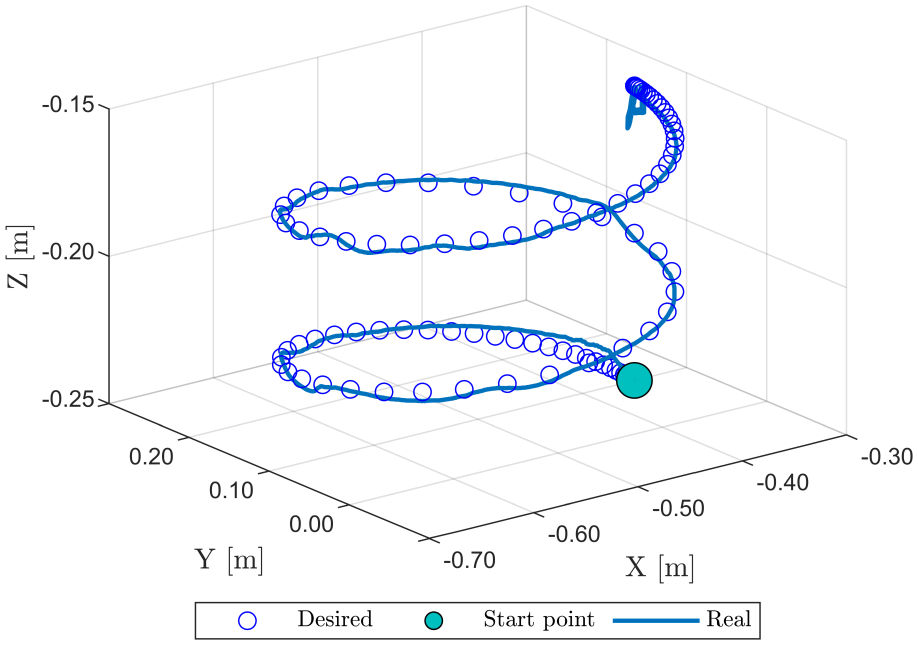


Figure 5.25: Experiment - Tool-tip trajectory.

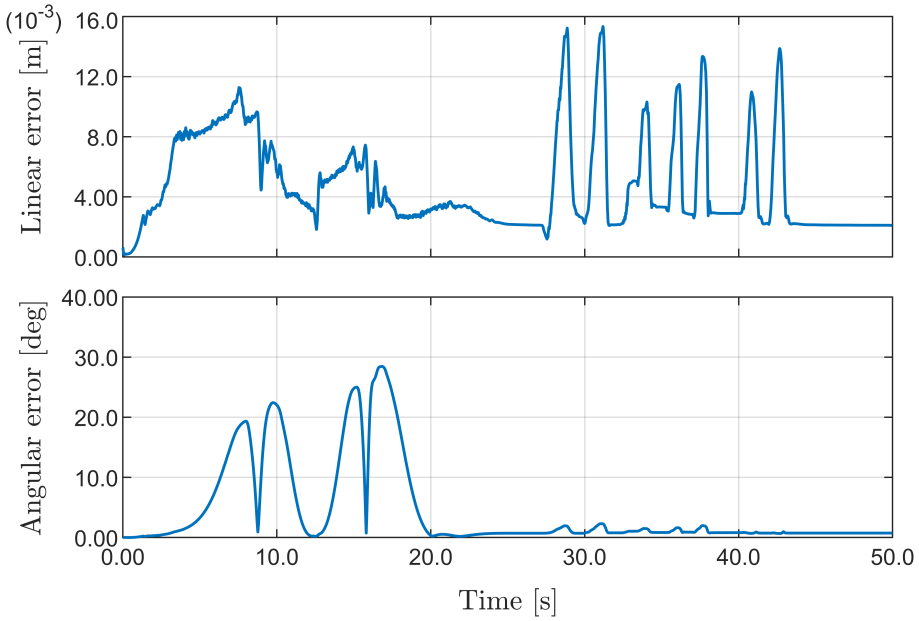


Figure 5.26: Experiment - Tool-tip trajectory tracking error norms.

Figures 5.25 and 5.26 reports the tool-tip trajectory and the tool-tip trajectory error norms of the experiment. An expected lack of accuracy in the tracking of the trajectory with respect to simulations is reported. Indeed, this is associated with the non-idealities of the real robot, the compliance control, and the inaccuracies of the model.

Figure 5.27 reports the RCM error absolute value along the three axes during the experiment. In the first $25s$, when the trajectory was performed without interaction, a maximum error norm of 1.4085mm is reported at $t = 15.692s$. This small error and the correct tracking of the trajectory prove the effectiveness of the proposed controller, and its capability to perform trajectory tracking while guaranteeing the RCM constraint.

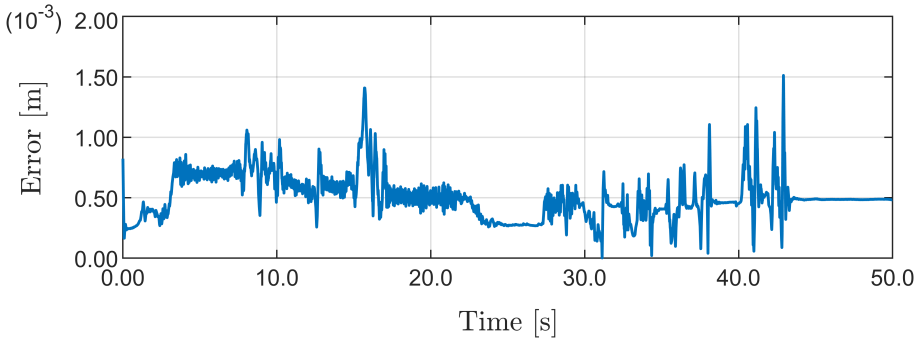


Figure 5.27: Experiment -RCM error norm.

Figure 5.28 reports the measured external forces and torques during the experiment. The interaction occurs at the end of the trajectory tracking, for $t > 25s$, when especially the forces notably increase. In the same time window, a notable deviation on the position of tool-tip is reported, as visible in Figure 5.26 for $t > 25s$. This does not holds for the RCM error. The movement of the tool-tip during the interaction and the associated low RCM error allows to state that the proposed controller is able to perform compliant motion control and effectively interact with the environment while guaranteeing the RCM constraint.

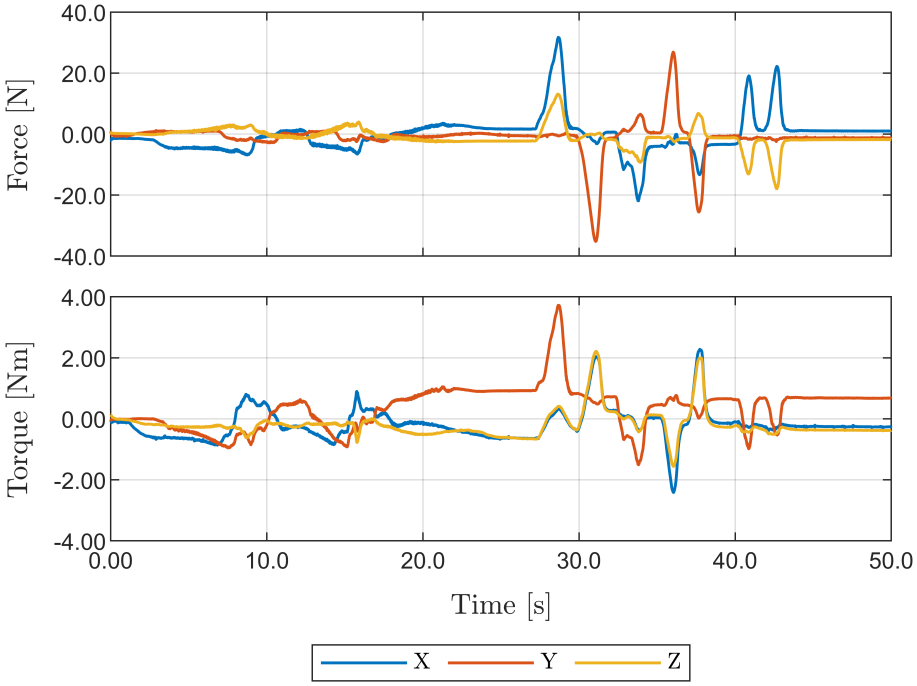


Figure 5.28: Experiment - Measured external forces and torques.

The RCM error was computed using the direct kinematic of the robot performed using the measurements of the encoders, and by using Equation (5.18). The RCM reports a very low maximum error norm on the entire experiment of $1.51mm$ at $t = 42.909$.

5.7 An RCM Compliant Bilateral Teleoperation Architecture

The controller proposed so far consider the dynamic of the overall constrained system and allows the user to design compliant motion control, making the system effectively interact with the environment.

Simulations and experiments results show that the proposed controller is able to perform a trajectory following task with different control actions (computed torque and impedance control action), guaranteeing the maintenance of the RCM with low error. Simulations and experiments show also the capability of the system to physically interact with the environment. Comparisons with other works show also that the proposed controller is capable of providing overall superior performance.

However, the parameters related to the RCM enforcement need to be remarkably changed when moving from simulations to experiments (see Equations (5.61) and (5.64)).

This is due to the fact that in simulation there are no differences between the robot model used in the controller and the simulated one. In these conditions, the error introduced on the RCM is mainly given by the finite integration time step with which the simulation is performed.

In the real case, the differences between the model used in the controller and the real

robot are greater, producing a remarkably higher error on the RCM, which needs to be compensated by raising the gains related to the RCM enforcement term.

To experimentally verify what is reported in the latter, and to demonstrate the versatility of the proposed controller, the extension of the controller to admittance controlled manipulator is proposed below. In fact, the implementation of admittance control allows to define the dynamic model of the controlled system a priori. This makes the controller work properly by using low gains in the RCM enforcement term.

Moreover, the performances of the controller are evaluated through the development of a bilateral smss teleoperation system.

5.8 Design of the Teleoperation System

5.8.1 Slave Side

Consider a n_s-DOFs admittance controlled manipulator represented by the following model:

$$\Lambda_s \ddot{x}_s(t) + \Pi_s \dot{x}_s(t) = F_s(t) + F_s^{ext}(t) \tag{5.65}$$

where $x_s \in \mathbb{R}^6$ is the Cartesian pose of the end-effector, $\Lambda_s \in \mathbb{R}^{6\times6}$ is the inertia matrix, $\Pi_s \in \mathbb{R}^{6\times6}$ is the damping matrix, $F_s \in \mathbb{R}^6$ is the Cartesian force control vector and $F_s^{ext} \in \mathbb{R}^6$ is the external Cartesian force vector.

The constraining method reported in Section 5.3.1 and used in Section 5.5 works at the joint torque level of the manipulator. This means that, in order to apply the same constraining method on (5.65), the equivalent model of (5.65) on the joint space needs to be computed. Assuming that the Jacobian matrix $J_s(q_s) \in \mathbb{R}^{6\times n}$ of the manipulator is full rank and far from singularity, this can be done by observing that:

$$\dot{x}_s = J_s(q)\dot{q}_s(t) \tag{5.66}$$

and:

$$\ddot{x}_s = J_s(q_s)\ddot{q}_s(t) + \dot{J}_s(q_s)\dot{q}_s(t) \tag{5.67}$$

where $q_s \in \mathbb{R}^{n_s}$ are the joint coordinate vector.

Substituting (5.66) and (5.67) in (5.65) leads to:

$$\Lambda_s J_s(q_s)\ddot{q}_s(t) + \Lambda_s \dot{J}_s(q_s)\dot{q}_s(t) + \Pi_s J_s(q_s)\dot{q}_s(t) = F_s(t) + F_s^{ext}(t) \tag{5.68}$$

which can be rewrite pre-multiplying both sides of the equation by $J_s(q_s)^T$ as:

$$J_s^T(q_s)\Lambda_s J_s(q_s)\ddot{q}_s(t) + J_s^T(q_s)\Lambda_s \dot{J}_s(q_s)\dot{q}_s(t) + J_s^T(q_s)\Pi_s J_s(q_s)\dot{q}_s(t) = J_s^T(q_s)\Big(F_s(t) + F_s^{ext}(t)\Big) \tag{5.69}$$

For sake of clarity, the dependences of the variables will be hereafter omitted when the context is clear.

Following the same principles used to constrain (5.1) in Section 5.5, the external force F_s^{ext} need first to be compensated to avoid undesired movement on the RCM due to uncontrolled external factors. This can be done by setting:

$$F_s = \bar{F}_s - F_s^{ext} \tag{5.70}$$

which leads to:

$$J_s^T(q_s)\Lambda_s J_s(q_s)\ddot{q}_s + J_s^T(q_s)\Lambda_s \dot{J}_s(q_s)\dot{q}_s + J_s^T(q_s)\Pi_s J_s(q_s)\dot{q}_s = J_s^T \bar{F}_s \tag{5.71}$$

By setting:

$$M_s^{eq}(q_s) = J_s^T(q_s)\Lambda_s J_s(q_s) \tag{5.72}$$

and:

$$\tau_s^{eq} = J_s^T(q_s)\bar{F}_s \tag{5.73}$$

the dynamic model of (5.65) in the joint space can be rewrite as:

$$M_s^{eq}(q_s)\ddot{q}_s + J_s^T(q_s)\Lambda_s \dot{J}_s(q_s)\dot{q}_s + J_s^T(q_s)\Pi_s J_s(q_s)\dot{q}_s = \tau_s^{eq} \tag{5.74}$$

with $\tau_s^{eq} \in \mathbb{R}^{n_s}$ the new control input. Once τ_s^{eq} has been defined, the real control input force F_s to be applied to the robot can be computed considering (5.73) and (5.70) as:

$$F_s = (J_s^T)^+ \tau_s^{eq} - F_s^{ext} \tag{5.75}$$

The model in (5.74) is not subject to any constraint. Once again, following the same principles used in Section 5.5 to constrain (5.1), the motion of the system (5.74) can be constrained to the RCM by setting:

$$\tau_s^{eq} = Q_s(q_s, \dot{q}_s) + Q_s^{constr}(q_s, \dot{q}_s) + J_s^T(q_s)\Lambda_s \dot{J}_s(q_s)\dot{q}_s + J_s^T(q_s)\Pi_s J_s(q_s)\dot{q}_s \tag{5.76}$$

where:

$$Q_s(q_s, \dot{q}_s) = \tau_s^\star - J_s^T(q_s)\Lambda_s \dot{J}_s(q_s)\dot{q}_s - J_s^T(q_s)\Pi_s J_s(q_s)\dot{q}_s \tag{5.77}$$

$$Q_s^{constr}(q_s, \dot{q}_s) = M_s^{eq}(q_s)^{\frac{1}{2}} B_s^+(q_s)\Big(b_{rcm}(q_s, \dot{q}_s) - A_{rcm}(q_s, \dot{q}_s) M_s^{eq}(q_s)^{-1} Q_s(q_s, \dot{q}_s)\Big) \tag{5.78}$$

$$B_s(q_s) = A_{rcm}(q_s, \dot{q}_s) M_s^{eq}(q_s)^{\frac{1}{2}} \tag{5.79}$$

with $\tau_s^\star \in \mathbb{R}^{n_s}$ the new control input.

Substituting (5.76) in (5.74) leads to:

$$M_s^{eq}(q_s)\ddot{q}_s = Q_s(q_s, \dot{q}_s) + Q_s^{constr}(q_s, \dot{q}_s) \tag{5.80}$$

that, with the same form of (5.9), is equivalent to the dynamic model of the manipulator (5.74) subject to the RCM constraint (5.40). This means that setting τ_s^{eq} as in (5.76) will make the system to behaves compliantly with the RCM, regardless of the choice of $\tau_s^\star$. As for the previous controller, the desired control action term $\tau_s^\star$ can be freely chosen by the user disregarding the RCM constraint. The term $\tau_s^\star$ can be then employed in (5.76)

to achieve the RCM compliant version of the desired control action, conferring extreme flexibility to the system.

Even if, thanks to admittance control, there are no differences between the dynamic model used inside the controller and that of the robot, the system still to be subject to problems related to the finite integration time of the controller. For this reason, an RCM enforcement term is added to the controller, as defined in (5.23) and reported in the following:

$$\tau^{rcm} = J_{rcm}^T \Big(K_{rcm}(x_{rcm}^{des} - x_{rcm}) - D_{rcm}\dot{x}_{rcm} \Big) \tag{5.81}$$

where $K_{rcm}, D_{rcm} \in \mathbb{R}^{3\times 3}$ are the proportional and derivative gains of the RCM error.

The overall controller can be then formulated as:

$$F_s = (J_s^T)^+ \Big(Q_s(q_s,\dot{q}_s) + Q_s^{constr}(q_s,\dot{q}_s) + J_s^T(q_s)\Lambda_s \dot{J}_s(q_s)\dot{q}_s + J_s^T(q_s)\Pi_s J_s(q_s)\dot{q}_s + \tau_{rcm} \Big) - F_s^{ext} \tag{5.82}$$

which make the system (5.65) behaves like:

$$M_s^{eq}(q_s)\ddot{q}_s = Q_s(q_s,\dot{q}_s) + Q_s^{constr}(q_s,\dot{q}_s) + \tau_{rcm} \tag{5.83}$$

with $\tau_s^\star$ in $Q_s(q,\dot{q})$ the new control input.

The admittance controlled constrained manipulator is a passive system, as proven in the following.

Proposition 5. *The system in* (5.65) *with the controller* (5.82) *is passive with respect to the pair* ($\tau_s^\star$, $\dot{q}_s$)

Proof. Consider as a storage function the total energy of the system in (5.65) with the controller (5.82), that is given by the sum of the kinetic energy associated to the inertia Λ_s and of the potential energy associated to the spring K_{rcm}:

$$V_s = \frac{1}{2}\dot{x}_s^T \Lambda_s \dot{x}_s + \frac{1}{2}\bar{x}_{rcm}^T K_{rcm} \bar{x}_{rcm} \tag{5.84}$$

where:

$$\bar{x}_{rcm} = x_{rcm}^{des} - x_{rcm} \tag{5.85}$$

Deriving (5.84) and considering that matrices Λ_s and K_{rcm} are constant leads to:

$$\dot{V}_s = \dot{x}_s^T \Lambda_s \ddot{x}_s + \bar{x}_{rcm}^T K_{rcm} \dot{\bar{x}}_{rcm} \tag{5.86}$$

Computing $\ddot{x}_s$ from (5.65) and using (5.82) and (5.77) returns:

$$\dot{V}_s = \dot{x}_s^T (J_s^T)^+ \tau_s^\star + \dot{x}_s^T (J_s^T)^+ Q_s^{constr} + \dot{x}_s^T (J_s^T)^+ \tau_{rcm} + \bar{x}_{rcm}^T K_{rcm} \dot{\bar{x}}_{rcm} \tag{5.87}$$

which, by setting:

$$\dot{V}_{s1} = \dot{x}_s^T (J_s^T)^+ \tau_s^\star \tag{5.88}$$

$$\dot{V}_{s2} = \dot{x}_s^T (J_s^T)^+ Q_s^{constr} \tag{5.89}$$

$$\dot{V}_{s3} = \dot{x}_s^T (J_s^T)^+ \tau_{rcm} + \bar{x}_{rcm}^T K_{rcm} \dot{\bar{x}}_{rcm} \tag{5.90}$$

becomes:

$$\dot{V}_s = \dot{V}_{s1} + \dot{V}_{s2} + \dot{V}_{s3} \tag{5.91}$$

The term (5.88) is the power flowing to/from the system due to the control action.

The term (5.89) is the product of the Cartesian velocity of the manipulator and the constraint Cartesian force, i.e. the virtual work. Since that force is computed using the Udwadia-Kalaba equation that represents the generalization of D'Alembert's principle, this term is equal to zero because of the principle of virtual work, namely:

$$\dot{V}_{s2} = 0 \tag{5.92}$$

Substituting (5.81) in (5.90) leads to:

$$\dot{V}_{s3} = \dot{x}_s^T (J_s^T)^+ J_{rcm}^T \Big(K_{rcm} \bar{x}_{rcm} - D_{rcm} \dot{x}_{rcm} \Big) + \bar{x}_{rcm}^T K_{rcm} \dot{\bar{x}}_{rcm} \tag{5.93}$$

which, observing that:

$$\dot{x}_{rcm} = J_{rcm} * J_s^+ \dot{x}_s \tag{5.94}$$

$$\Big((J_s^T)^+ \Big)^T = J_s^+ \tag{5.95}$$

$$K_{rcm}^T = K_{rcm} \tag{5.96}$$

and:

$$\dot{\bar{x}}_{rcm} = -\dot{x}_{rcm} \tag{5.97}$$

becomes:

$$\dot{V}_{s3} = -\dot{x}_s^T (J_s^T)^+ J_{rcm}^T D_{rcm} J_{rcm} J_s^+ \dot{x}_s \tag{5.98}$$

which is always negative.

Equation (5.87) can be then rewrite as:

$$\dot{V}_s = \dot{x}_s^T (J_s^T)^+ \tau_s^\star - \dot{x}_s^T (J_s^T)^+ J_{rcm}^T D_{rcm} J_{rcm} J_s^+ \dot{x}_s \tag{5.99}$$

Since:

$$\dot{x}_s^T (J_s^T)^+ J_{rcm}^T D_{rcm} J_{rcm} J_s^+ \dot{x}_s > 0 \tag{5.100}$$

it follows that:

$$\dot{V}_s \leq \dot{x}_s^T (J_s^T)^+ \tau_s^\star \tag{5.101}$$

and since:

$$\dot{x}_s^T (J_s^T)^+ = \dot{q}_s^T \tag{5.102}$$

the following passivity condition holds:

$$\int_0^t \dot{q}_s^T(\zeta) \tau_s^\star(\zeta) d\zeta \geq -V_s(0) \tag{5.103}$$

□

5.8.2 Master Side

The master device is used by the operator, and send to the slave device position and velocity information. In order to leave the operator free to perform the desired movements, no controller is applied to enforce the RCM.

Thus, the master side is considered as a n_m-DOFs admittance controlled manipulator represented by the following model:

$$\Lambda_m \ddot{x}_m(t) + \Pi_m \dot{x}_m(t) = F_m(t) + F_m^{ext}(t) \tag{5.104}$$

where $x_m \in \mathbb{R}^6$ is the Cartesian pose of the end-effector, $\Lambda_m \in \mathbb{R}^{6\times 6}$ is the inertia matrix, $F_m \in \mathbb{R}^6$ is the Cartesian force control vector and $F_m^{ext} \in \mathbb{R}^6$ is the external Cartesian force vector.

Both master and slave will be augmented with a tank. In order to develop the same tank for both master and slave side, and since the slave controller works at the joint torque level, the model of the master manipulator at the joint space is considered.

Following the same steps that lead from (5.65) to (5.69), the master manipulator model at the joint space can be computed as:

$$J_m^T \Lambda_m J_m \ddot{q}_m + J_m^T \Lambda_m \dot{J}_m \dot{q}_m + J_m^T \Pi_m J_m \dot{q}_m = J_m^T \Big(F_m + F_m^{ext} \Big) \tag{5.105}$$

By setting:

$$M_m^{eq} = J_m^T \Lambda_m J_m \tag{5.106}$$

and:

$$\tau_m^{eq} = J_m^T F_m \tag{5.107}$$

the dynamic model of (5.104) in the joint space can be rewritten as:

$$M_m^{eq} \ddot{q}_m + J_m^T \Lambda_m \dot{J}_m \dot{q}_m + J_m^T \Pi_m J_m \dot{q}_m = \tau_m^{eq} + J_m^T F_m^{ext} \tag{5.108}$$

with $\tau_m^{eq} \in \mathbb{R}^{n_m}$ the new control input. Once τ_m^{eq} has been defined, the real control input force F_m to be applied to the robot can be computed considering (5.107) as:

$$F_m = (J_m^T)^+ \tau_m^{eq} \tag{5.109}$$

The admittance controlled manipulator is a passive system, as proven in the following.

Proposition 6. *The system in* (5.108) *is passive with respect to the pairs* (τ_m^{eq}, $\dot{q}_m$) *and* (F_m^{ext}, $\dot{x}_m$)

Proof. Consider as a storage function the total energy of the system in (5.108), which is given by the kinetic energy associated to the inertia Λ_m:

$$V_m = \frac{1}{2} \dot{x}_m^T \Lambda_m \dot{x}_m \tag{5.110}$$

Deriving (5.110) and considering that the matrix Λ_m is constant leads to:

$$\dot{V}_m = \dot{x}_m^T \Lambda_m \ddot{x}_m \tag{5.111}$$

Computing $\ddot{x}_m$ from (5.104) and using (5.109) returns:

$$\dot{V}_m = \dot{x}_m^T (J_m^T)^+ \tau_m^{eq} + \dot{x}_m^T F_m^{ext} - \dot{x}_m^T \Pi_m \dot{x}_m \tag{5.112}$$

Since:

$$\dot{x}_m^T \Pi_m \dot{x}_m > 0 \tag{5.113}$$

it follows that:

$$\dot{V}_m \leq \dot{x}_m^T (J_m^T)^+ \tau_m^{eq} + \dot{x}_m^T F_m^{ext} \tag{5.114}$$

and since:

$$\dot{x}_m^T (J_m^T)^+ = \dot{q}_m^T \tag{5.115}$$

the following passivity condition holds:

$$\int_0^t \dot{q}_m^T(\zeta) \tau_m^{eq}(\zeta) + \dot{x}_m^T(\zeta) F_m^{ext}(\zeta) d\zeta \geq -V_m(0) \tag{5.116}$$

□

5.8.3 The Teleoperation Architecture

Since the master and slave manipulators will be employed on a teleoperation setup, the passivity of the controlled manipulators does not guarantee the passivity of the overall system, especially when destabilizing factors like interaction with a poorly known environment or communication delays occur.

For this reason, both master and slave sides are augmented with a tank. Inspiring on the formulation of the shared energy tank reported in Section 2.1.4, the control inputs of the controlled manipulators are split into the sum of two terms:

$$\begin{cases} \tau_m^{eq} &= \tau_m^{\tau} + \tau_m^d \\ \tau_s^{\star} &= \tau_s^{\tau} + \tau_s^d \end{cases} \tag{5.117}$$

The first term is used to implement a control action on the manipulator while the second term is exploited for implementing a variable local damping by setting:

$$\begin{cases} \tau_m^d = -D_m(t) \dot{q}_m(t) \\ \tau_s^d = -D_s(t) \dot{q}_s(t) \end{cases} \tag{5.118}$$

where $D_m(t) \in \mathbb{R}^{n_m \times n_m}$ and $D_s(t) \in \mathbb{R}^{n_s \times n_s}$ are time-varying positive semi-definite matrices. By considering the dynamic model of the controlled manipulators at the joint space, and by embedding the damping injection (5.117) and (5.118) the following damped

models for the manipulators can be computed:

$$\begin{cases} M_s^{eq}\ddot{q}_s = \mathscr{Q}_s(q_s,\dot{q}_s) + \mathscr{Q}_s^{constr}(q_s,\dot{q}_s) + \tau_{rcm} \\ M_m^{eq}\ddot{q}_m + J_m^T\Lambda_m\dot{J}_m\dot{q}_m + J_m^T\Pi_m J_m\dot{q}_m + D_m(t)\dot{q}_m(t) = \tau_m^\tau + + J_m^T F_m^{ext} \end{cases} \tag{5.119}$$

with:

$$\mathscr{Q}_s(q_s,\dot{q}_s) = \tau_s^\tau - D_s(t)\dot{q}_s - J_s^T(q_s)\Lambda_s\dot{J}_s(q_s)\dot{q}_s - J_s^T(q_s)\Pi_s J_s(q_s)\dot{q}_s \tag{5.120}$$

and:

$$\mathscr{Q}_s^{constr}(q_s,\dot{q}_s) = M_s^{eq}(q_s)^{\frac{1}{2}} B_s^+(q_s)\Big(b_{rcm}(q_s,\dot{q}_s) - A_{rcm}(q_s,\dot{q}_s)M_s^{eq}(q_s)^{-1}\mathscr{Q}_s(q_s,\dot{q}_s)\Big) \tag{5.121}$$

An energy tank is then placed at both master and slave sides, which model is represented by:

$$\begin{cases} \dot{x}_{t_w} = \dfrac{\dot{q}_w^T D_w(t)\dot{q}_w + \sigma_w P_w^{in}}{x_{t_w}} + u_{t_w} - \dfrac{P_w^{out}}{x_{t_w}} \\ y_{t_w} = \dfrac{\partial T_w}{\partial x_{t_w}} \end{cases} \tag{5.122}$$

with $w \in \{m, s\}$ where the subscripts m and s are used to indicate the master side and the slave side respectively.

The term $x_{t_w} \in \mathbb{R}$ is the state of the tank and $(u_{t_w}, y_{t_w}) \in \mathbb{R} \times \mathbb{R}$ is the power port through which the tank can exchange energy with the manipulator. The ports P_w^{in} and P_w^{out} are the input and output ports through which the tank can exchange energy with the other side of the teleoperation system.

The energy stored into the tank can be computed as:

$$T_w(x_{t_w}) = \frac{1}{2}x_{t_w}^2 \tag{5.123}$$

Using (5.123) with (5.122) it is easy to see that:

$$\dot{T}_w(x_{t_w}) = \dot{q}_w^T D_w(t)\dot{q}_w + \sigma_w P_w^{in} + u_{t_w}y_{t_w} - P_w^{out} \tag{5.124}$$

namely, the tanks stores the energies dissipated by the robots using the local damping injection and that energies can be injected/extracted from the ports (u_{t_w}, y_{t_w}), P_w^{in} and P_w^{out}.

Each robot is then interconnected to its energy tanks in order to exploit the energy stored into the tanks for implementing the desired action. This can be done by setting the following power preserving interconnections:

$$\tau_w^\tau = \omega_w y_{t_w} \tag{5.125}$$

Since:

$$\dot{q}_w^T\tau_w^\tau = -u_{t_w}y_{t_w} \tag{5.126}$$

it means that each robot can extract/inject energy from/in its tanks in order to implement the desired action by properly choosing the modulation factor $\omega_w \in \mathbb{R}^{n_w}$.

The terms σ_w is defined using the improved energy tank as in (2.57), which guarantees a more effective use of energy inside the teleoperation system. All the considerations reported in Section 2.1.4 about the bounding on the maximum and minimum energy stored into the tank and on the design of the damping term D_w still hold.

Considering (5.119), (5.122), and (5.125) it is possible to model the overall system as:

$$\begin{cases} M_m^{eq}\ddot{q}_m + J_m^T\Lambda_m\dot{J}_m\dot{q}_m + J_m^T\Pi_m J_m\dot{q}_m + D_m(t)\dot{q}_m(t) = \omega_m x_{t_m} \\ \dot{x}_{t_m} = \dfrac{\dot{q}_m^T D_m(t)\dot{q}_m + \sigma_m P_m^{in}}{x_{t_m}} - \omega_m^T\dot{q}_m + \dfrac{P_m^{out}}{x_{t_m}} \\ M_s^{eq}(q_s)\ddot{q}_s = {}^{\omega}\mathscr{Q}_s(q_s,\dot{q}_s) + {}^{\omega}\mathscr{Q}_s^{constr}(q_s,\dot{q}_s) + \tau_{rcm} \\ \dot{x}_{t_s} = \dfrac{\dot{q}_s^T D_s(t)\dot{q}_s + \sigma_s P_s^{in}}{x_{t_s}} - \omega_s^T\dot{q}_s + \dfrac{P_s^{out}}{x_{t_s}} \end{cases} \tag{5.127}$$

with:

$${}^{\omega}\mathscr{Q}_s(q_s,\dot{q}_s) = \omega_s x_{t_s} - D_s(t)\dot{q}_s - J_s^T(q_s)\Lambda_s\dot{J}_s(q_s)\dot{q}_s - J_s^T(q_s)\Pi_s J_s(q_s)\dot{q}_s \tag{5.128}$$

and:

$${}^{\omega}\mathscr{Q}_s^{constr}(q_s,\dot{q}_s) = M_s^{eq}(q_s)^{\frac{1}{2}}B_s^{+}(q_s)\Big(b_{rcm}(q_s,\dot{q}_s) - A_{rcm}(q_s,\dot{q}_s)M_s^{eq}(q_s)^{-1}\,{}^{\omega}\mathscr{Q}_s(q_s,\dot{q}_s)\Big) \tag{5.129}$$

The desired input for each robots can be achieved by setting the modulating terms as:

$$\omega_w = \begin{cases} \dfrac{\tau_w^{des}}{x_{t_w}} & \text{if } T(x_{t_w}) \geq T_w^R \\ K_w(T(x_w))\dfrac{\tau_w^{des}}{x_{t_w}} & \text{otherwise} \end{cases} \tag{5.130}$$

with:

$$K_w(T(x_{t_w})) = \max\left(0, \frac{T(x_{t_w}) - T_w^{min}}{T_w^R - T_w^{min}}\right) \tag{5.131}$$

Thus, if there is enough energy in the tank, the desired input is implemented, otherwise, only a scaled version of the desired input is implemented. In the worst case $K_w(T(x_w)) = 0$ and, therefore, nothing will be implemented in order to preserve passivity. Nevertheless, since the local damping is set to its maximum when $T(x_{t_w}) < T_w^R$, its very unlikely that $K_w(T(x_w)) = 0$ in practice.

The policy used for defining P_m^{out} and P_s^{out} in (5.127) is the same as (2.35).

In presence of time delay Δt between the two sides:

$$\begin{cases} P_s^{in}(t) = P_m^{out}(t - \Delta t) \\ P_m^{in}(t) = P_s^{out}(t - \Delta t) \end{cases} \tag{5.132}$$

which turns the communication channel into an energy-storing element. In particular:

$$H_{ch}(t) = \int_{t-\Delta t}^{t} P_m^{out}(\tau) + P_s^{out}(\tau) d\tau \tag{5.133}$$

where $H_{ch}(t)$ is the energy stored in the communication channel, and:

$$\dot{H}_{ch}(t) = P_m^{out}(t) - P_m^{out}(t-\delta) + P_s^{out}(t) - P_s^{out}(t-\delta) \tag{5.134}$$

The system in (5.127) consists of the master manipulator (5.83) with the damping element D_m and the master tank that is energetically coupled through the input ω_m, and of the slave manipulator (5.83) with the damping element D_s and the slave tank that is energetically coupled through the input ω_s.

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

Proposition 7. *The system in* (5.127) *is passive with respect to the pair* (F_m^{ext},$\dot{x}_m$).

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

$$V = V_m + T_m + V_s + T_s + H_{ch} \tag{5.135}$$

Using the same procedure that leads from (5.110) to (5.112), for the master manipulator coupled with the tank, it follows that:

$$\dot{V}_m = \dot{x}_m^T (J_m^T)^+ \omega_m x_{t_m} + \dot{x}_m^T F_m^{ext} - \dot{x}_m^T \Pi_m \dot{x}_m - \dot{x}_m^T (J_m^T)^+ D_m J_m^+ \dot{x}_m \tag{5.136}$$

while using the same procedure that leads from (5.84) to (5.99), for the slave manipulator coupled with the tank, leads to:

$$\dot{V}_s = \dot{x}_s^T (J_s^T)^+ \omega_s x_{t_s} - \dot{x}_s^T (J_s^T)^+ D_s(t) J_s^+ \dot{x}_s - \dot{x}_s^T (J_s^T)^+ J_{rcm}^T D_{rcm} J_{rcm} J_s^+ \dot{x}_s \tag{5.137}$$

Furthermore, by plugging (5.126) and (5.125) in (5.124) it follows that:

$$\dot{T}_m = \dot{q}_m^T D_m \dot{q}_m + \sigma_m P_m^{in} - x_{t_m} \omega_m^T \dot{q}_m - P_w^{out} \tag{5.138}$$

and:

$$\dot{T}_s = \dot{q}_s^T D_s \dot{q}_s + \sigma_s P_s^{in} - x_{t_s} \omega_s^T \dot{q}_s - P_s^{out} \tag{5.139}$$

which can be rewrite as:

$$\dot{T}_m = \dot{x}_m^T (J_m^+)^T D_m J_m^+ \dot{x}_m - x_{t_m} \omega_m^T J_m^+ \dot{x}_m + \sigma_m P_m^{in} - P_m^{out} \tag{5.140}$$

and:

$$\dot{T}_s = \dot{x}_s^T (J_s^+)^T D_s J_s^+ \dot{x}_s - x_{t_s} \omega_s^T J_s^+ \dot{x}_s + \sigma_s P_s^{in} - P_s^{out} \tag{5.141}$$

Deriving (5.135) and using (5.136), (5.137), (5.140), (5.141) and (5.134) allows to obtain that:

$$\begin{aligned}\dot{V} = \dot{x}_m^T F_m^{ext} - \dot{x}_m^T \Pi_m \dot{x}_m - \dot{x}_s^T (J_s^T)^+ J_{rcm}^T D_{rcm} J_{rcm} J_s^+ \dot{x}_s + \\ + \sigma_m P_m^{in} - P_m^{out} + \sigma_s P_s^{in} - P_s^{out} + \\ P_m^{out}(t) - P_m^{out}(t-\delta) + P_s^{out}(t) - P_s^{out}(t-\delta)\end{aligned} \tag{5.142}$$

Using (5.132) leads to:

$$\begin{aligned}\dot{V} = \dot{x}_m^T F_m^{ext} - \dot{x}_m^T \Pi_m \dot{x}_m - \dot{x}_s^T (J_s^T)^+ J_{rcm}^T D_{rcm} J_{rcm} J_s^+ \dot{x}_s + \\ - (1-\sigma_m) P_m^{in} - (1-\sigma_s) P_s^{in}\end{aligned} \tag{5.143}$$

In virtue of (5.100) and (5.113), since $\sigma_w \in \{0,1\}$, from Lemma 1 $P_w^{in} \geq 0$, it follows that:

$$\dot{V} \leq \dot{x}_m^T F_m^{ext} \tag{5.144}$$

which imply the following passivity condition:

$$\int_0^t \dot{x}_m^T(\zeta) F_m^{ext}(\zeta) d\zeta \geq -V(0) \tag{5.145}$$

□

5.9 Validation

In order to prove the effectiveness of the proposed teleoperation architecture, a KUKA LWR 4+ 7-DOF robot with a laparoscopic-like tool mounted at the end-effector is used at the slave side and a Universal Robot UR10e with a laparoscopic-like joystick mounted at the end-effector is used at the master side as haptic interface.

The following desired control actions are set:

$$\begin{cases}\tau_m^{des} = J_m^T (J_s^T)^+ \, {}^{\omega}\mathcal{Q}_s^{constr} \\ \tau_s^{des} = J_s^T \Big(K_p(x_m - x_s) + K_d(\dot{x}_m - \dot{x}_s) \Big)\end{cases} \tag{5.146}$$

In order to give the user the feeling of being constrained on the movement of the slave, at the master side the constrain force of the slave manipulator is fed back. At the slave side, a PD control action is set to allow the slave tracking the master movements.

The following parameters were also set:

$$\begin{aligned}\Lambda^m = \Lambda^s = \text{diag}(2.5, 2.5, 2.5, 0.05, 0.05, 0.05) \\ \Pi^m = \Pi^s = \text{diag}(5, 5, 5, 0.3, 0.3, 0.3) \\ K_{rcm} = \text{diag}(10.0, 10.0, 10.0) \\ D_{rcm} = \text{diag}(2.5, 2.5, 2.5)\end{aligned} \tag{5.147}$$

with Table 5.5 reporting the parameters used for the tanks.

Parameter	Master side	Slave side	Unit
T_w^{max}	5.0	5.0	J
T_w^{min}	0.1	0.1	J
$T_w^{b_{max}}$	3.0	3.0	J
$T_w^{b_{min}}$	1.5	1.5	J
T_w^R	1.5	1.5	J
T_w^{ava}	3.0	3.0	J
T_w^{req}	1.5	1.5	J
$\bar{P}$	0.25	0.25	J

Table 5.5: Controller parameters used for the experimental evaluation.

The experiment is performed by moving the master device to replicate a surgical procedure, and evaluating the tracking performance.

Figure 5.29 reports the trajectory tracking error norms. It can be observed that the norm of the linear error is kept at a low value during the whole experiment, while the norm of the angular error reports much higher values. This is an expected result, as the controller tries to keep the RCM position fixed, modifying the orientations of the desired trajectory.

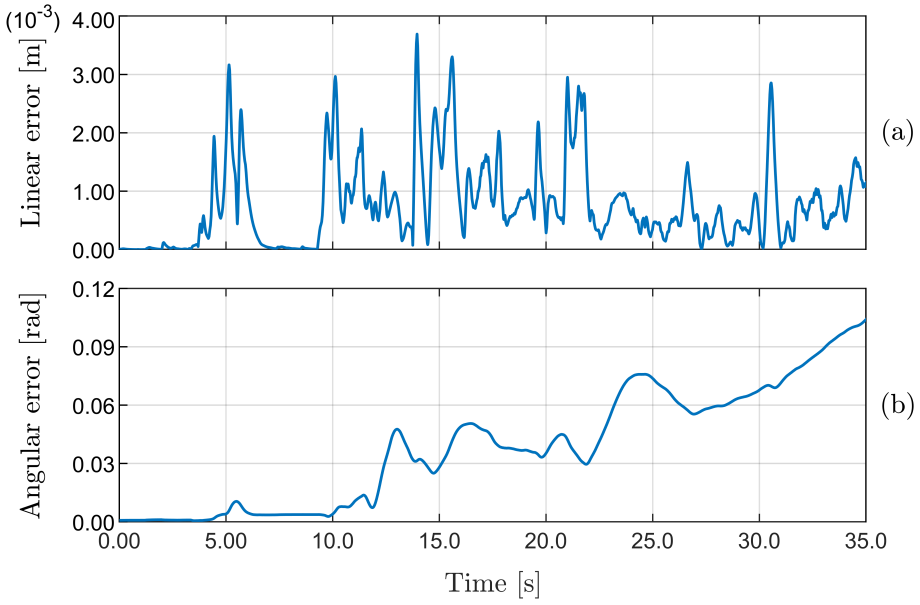


Figure 5.29: Trajectory tracking error norms.

This behavior is confirmed by looking at the Figure 5.30, which shows the error norm of the RCM. it can be observed that this value is kept close to the value of $1mm$ for most of the duration of the experiment, comparable with the one obtained during the validation of the proposed controller in the case of impedance control.

Finally, Figure 5.31 reports the energy dissipated by the manipulators and stored in the master and slave tanks. Starting from an initial value of $1J$, the energy grows continuously. This is due to the introduction of local damping, which dissipates more energy than is required to implement the desired control action. This allows the system to operate correctly.

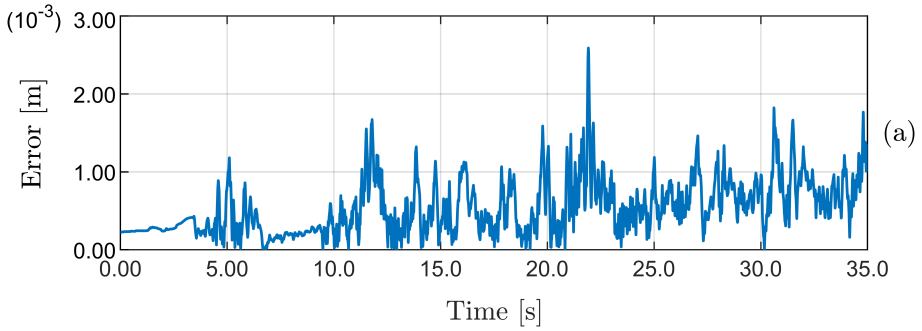


Figure 5.30: RCM tracking error norm.

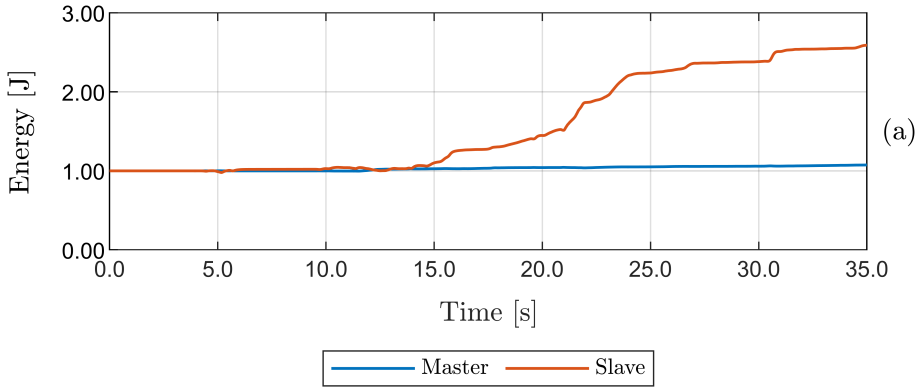


Figure 5.31: Energy stored into the master and slave tank

5.10 Conclusions

This chapter deals with the development of a torque controller able of constraining the motion of a manipulator to the RCM constraint. The equivalent dynamic model of the manipulator system subject to the constraint is computed and exploited to synthesize the controller. Moreover, the controller allows the user to freely specify the desired input disregarding the constraint.

The first controller was based on the calculation of virtual joints and the emulation of a virtual dynamics demonstrated the effectiveness of the strategy, but it was unnecessarily complicated, making difficult for the system to interact effectively with the environment.

A second development reports a considerable improvement, allowing the system to implement different control actions, including impedance control, through which it was possible to demonstrate the capability of the system to interact effectively with the environment.

On both controllers, it was reported the need to insert an enforcement term on the position of the RCM, especially when moving from simulations to practical experiments, where the real model of the robot is not always reliable.

With the latter in mind, the strategy used for both controllers was extended to the admittance control, and to a validation within a teleoperation system.

However, to achieve high performance it was not possible to remove the use of the enforcement term on the RCM, but the gains used compared to previous versions further confirm the results obtained.

Future works aim at developing a new formulation of the RCM constraint capable of considering the aforementioned problem, making the overall controller work without the RCM enforcement term.

Chapter 6

Conclusions and Future Directions

The work of this thesis addressed some of the challenging problems related to the control of surgical robots when moving from a teleoperated system to an autonomous system. The problems addressed are just some of those that affect this field and were mainly those that emerged in the development of the SARAS project, with some exceptions for the assistive strategies presented in the field of training of novice surgeons.

Strategies for the control of teleoperation systems aimed both at the acquisition of data for the training of AI systems, and for the assistance and training of novice surgeon was treated at the beginning of this work. New concepts like the shared energy tank have been proposed and implemented on different systems, reporting excellent performance and high flexibility of use, regardless of the number of agents within the teleoperation system.

The problem of coordinating several autonomous laparoscopic instruments has been addressed in Chapter 3, developing and optimizing a coordination system that allows the instruments to work in the same workspace and to work together with a teleoperated system. The proposed system allows completing the task safely, avoiding collisions with the other tools in the workspace, and dealing with the uncertainty related to the action requested by a cognitive module.

In the field of training of novice surgeons, a robotic assistance architecture that can assist the surgeon during renal access in PCNL procedures is also proposed. Results with multiple users show the advantages of using robot assistance.

Finally, the problem of controlling the RCM with a light-weight torque-controlled robot has been addressed, proposing a new control strategy based on the dynamic model of the constrained system. Multiple development steps have made possible to synthesize a flexible controller, able to interact effectively with the environment, and able to be embedded in a teleoperation architecture. This development aims to offer a low-cost alternative to expensive laparoscopic teleoperation systems that may be not affordable in many cases.

The latest developments in AI report extremely promising results. However, these systems are unable to provide guarantees and are not very robust. In the field of robotic

surgery, where patient safety is the holy grail, it places various limits on the application of these systems. From my experience gained in this PhD, the big step to take in terms of control is to merge the potentiality of AI systems into frameworks capable of returning guarantees.

Bibliography

[1] M. Minelli, N. Piccinelli, F. Falezza, F. Ferraguti, R. Muradore, and C. Secchi, "Two-layer based multi-arms bilateral teleoperation architecture," Submitted to IEEE Transactions on Control System Technology.

[2] A. Leporini, E. Oleari, C. Landolfo, A. Sanna, A. Larcher, G. Gandaglia, N. Fossati, F. Muttin, U. Capitanio, F. Montorsi, *et al.*, "Technical and functional validation of a teleoperated multirobots platform for minimally invasive surgery," *IEEE Transactions on Medical Robotics and Bionics*, vol. 2, no. 2, pp. 148–156, 2020.

[3] M. Minelli, F. Loschi, F. Ferraguti, and C. Secchi, "A two-layer trilateral teleoperation architecture for mentoring in surgical training," *IFAC-PapersOnLine*, vol. 54, no. 19, pp. 267–274, 2021.

[4] G. De Rossi, M. Minelli, S. Roin, F. Falezza, A. Sozzi, F. Ferraguti, F. Setti, M. Bonfè, C. Secchi, and R. Muradore, "A first evaluation of a multi-modal learning system to control surgical assistant robots via action segmentation," *IEEE Transactions on Medical Robotics and Bionics*, vol. 3, no. 3, pp. 714–724, 2021.

[5] F. Ferraguti, M. Minelli, S. Farsoni, S. Bazzani, M. Bonfè, A. Vandanjon, S. Puliatti, G. Bianchi, and C. Secchi, "Augmented reality and robotic-assistance for percutaneous nephrolithotomy," *IEEE robotics and automation letters*, vol. 5, no. 3, pp. 4556–4563, 2020.

[6] S. Puliatti, M. Amato, F. Ferraguti, M. Minelli, S. Farsoni, A. Eissa, M. Rizzo, L. Bevilacqua, M. Sighinolfi, C. Secchi, *et al.*, "A combined augmented reality and robotic system for assistance in percutaneous nephrolithotomy procedures," *European Urology Open Science*, vol. 32, p. S44, 2021.

[7] ——, "A combined augmented reality and robotic system for assistance in percutaneous nephrolithotomy procedures," *European Urology Open Science*, vol. 20, p. S70, 2020.

[8] M. Minelli and C. Secchi, "A torque controlled approach for virtual remote centre of motion implementation," Submitted to IEEE robotics and automation letters.

[9] M. Amato, A. Eissa, S. Puliatti, C. Secchi, F. Ferraguti, M. Minelli, A. Meneghini, I. Landi, G. Guarino, M. C. Sighinolfi, *et al.*, "Feasibility of a telementoring approach as a practical training for transurethral enucleation of the benign prostatic hyperplasia using bipolar energy: a pilot study," *World journal of urology*, vol. 39, no. 9, pp. 3465–3471, 2021.

[10] M. Minelli, F. Ferraguti, N. Piccinelli, R. Muradore, and C. Secchi, "An energy-shared two-layer approach for multi-master-multi-slave bilateral teleoperation systems," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 423–429.

[11] M. Minelli, A. Sozzi, G. De Rossi, F. Ferraguti, F. Setti, R. Muradore, M. Bonfè, and C. Secchi, "Integrating model predictive control and dynamic waypoints generation for motion planning in surgical scenario," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 3157–3163.

[12] M. Minelli, A. Sozzi, G. De Rossi, F. Ferraguti, S. Farsoni, F. Setti, R. Muradore, M. Bonfè, and C. Secchi, "Linear mpc-based motion planning for autonomous surgery," Submitted to 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2020.

[13] G. De Rossi, M. Minelli, A. Sozzi, N. Piccinelli, F. Ferraguti, F. Setti, M. Bonfé, C. Secchi, and R. Muradore, "Cognitive robotic architecture for semi-autonomous execution of manipulation tasks in a surgical environment," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 7827–7833.

[14] M. Minelli and C. Secchi, "Dynamic-based rcm torque controller for robotic-assisted minimally invasive surgery," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 733–740.

[15] J. Panerati, M. Minelli, C. Ghedini, L. Meyer, M. Kaufmann, L. Sabattini, and G. Beltrame, "Robust connectivity maintenance for fallible robots," *Autonomous Robots*, vol. 43, no. 3, pp. 769–787, 2019.

[16] L. Siligardi, J. Panerati, M. Kaufmann, M. Minelli, C. Ghedini, G. Beltrame, and L. Sabattini, "Robust area coverage with connectivity maintenance," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2202–2208.

[17] M. Minelli, M. Kaufmann, J. Panerati, C. Ghedini, G. Beltrame, and L. Sabattini, "Stop, think, and roll: Online gain optimization for resilient multi-robot topologies," in *Distributed Autonomous Robotic Systems*. Springer, 2019, pp. 357–370.

[18] M. Minelli, J. Panerati, M. Kaufmann, C. Ghedini, G. Beltrame, and L. Sabattini, "Self-optimization of resilient topologies for fallible multi-robots," *Robotics and Autonomous Systems*, vol. 124, p. 103384, 2020.

[19] 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*, vol. 35, no. 2, pp. 153–160, 1988.

[20] H. A. Paul, W. L. Bargar, B. Mittlestadt, B. Musits, R. H. Taylor, P. Kazanzides, J. Zuhars, B. Williamson, and W. Hanson, "Development of a surgical robot for cementless total hip arthroplasty," *Clinical Orthopaedics and related research*, no. 285, pp. 57–66, 1992.

[21] "United states food and drug administration," https://www.fda.gov.

[22] S. Unger, H. Unger, and R. Bass, "Aesop robotic arm," *Surgical endoscopy*, vol. 8, no. 9, pp. 1131–1131, 1994.

[23] H. Reichenspurner, R. J. Damiano, M. Mack, D. H. Boehm, H. Gulbins, C. Detter, B. Meiser, R. Ellgass, and B. Reichart, "Use of the voice-controlled and computer-assisted surgical system zeus for endoscopic coronary artery bypass grafting," *The Journal of thoracic and cardiovascular surgery*, vol. 118, no. 1, pp. 11–16, 1999.

[24] "Intuitive surgical," https://www.intuitive.com/en-us.

[25] C. Freschi, V. Ferrari, F. Melfi, M. Ferrari, F. Mosca, and A. Cuschieri, "Technical review of the da vinci surgical telemanipulator," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 9, no. 4, pp. 396–406, 2013.

[26] J. H. Palep, "Robotic assisted minimally invasive surgery," *Journal of minimal access surgery*, vol. 5, no. 1, p. 1, 2009.

[27] Y.-M. Wu, R.-H. Hu, H.-S. Lai, and P.-H. Lee, "Robotic-assisted minimally invasive liver resection," *Asian journal of surgery*, vol. 37, no. 2, pp. 53–57, 2014.

[28] G. I. van Boxel, B. F. Kingma, F. J. Voskens, J. P. Ruurda, and R. van Hillegersberg, "Robotic-assisted minimally invasive esophagectomy: past, present and future," *Journal of Thoracic Disease*, vol. 12, no. 2, p. 54, 2020.

[29] A. Chiu, W. B. Bowne, K. A. Sookraj, M. E. Zenilman, A. Fingerhut, and G. S. Ferzli, "The role of the assistant in laparoscopic surgery: important considerations for the apprentice-in-training," *Surgical innovation*, vol. 15, no. 3, pp. 229–236, 2008.

[30] J. A. Wahr, R. L. Prager, J. Abernathy III, E. A. Martinez, E. Salas, P. C. Seifert, R. C. Groom, B. D. Spiess, B. E. Searles, T. M. Sundt III, *et al.*, "Patient safety in the cardiac operating room: human factors and teamwork: a scientific statement from the american heart association," *Circulation*, vol. 128, no. 10, pp. 1139–1169, 2013.

[31] "World healt organization," https://www.who.int.

[32] J. Ren, R. V. Patel, K. A. McIsaac, G. Guiraudon, and T. M. Peters, "Dynamic 3-d virtual fixtures for minimally invasive beating heart procedures," *IEEE transactions on medical imaging*, vol. 27, no. 8, pp. 1061–1070, 2008.

[33] S. Chang, J. Kim, I. Kim, J. H. Borm, C. Lee, and J. O. Park, "Kist teleoperation system for humanoid robot," in *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No. 99CH36289)*, vol. 2. IEEE, 1999, pp. 1198–1203.

[34] J. Scholtz, M. Theofanos, and B. Antonishek, "Development of a test bed for evaluating human-robot performance for explosive ordnance disposal robots," in *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, 2006, pp. 10–17.

[35] F. Ferraguti, N. Preda, A. Manurung, M. Bonfe, O. Lambercy, R. Gassert, R. Muradore, P. Fiorini, and C. Secchi, "An energy tank-based interactive control architecture for autonomous and teleoperated robotic surgery," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1073–1088, 2015.

[36] D. A. Lawrence, "Stability and transparency in bilateral teleoperation," *IEEE transactions on robotics and automation*, vol. 9, no. 5, pp. 624–637, 1993.

[37] 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*, vol. 27, no. 4, pp. 741–756, 2011.

[38] G. Niemeyer, C. Preusche, S. Stramigioli, and D. Lee, "Telerobotics," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer Berlin Heidelberg, 2016, ch. 42.

[39] G. Niemeyer and J.-J. E. Slotine, "Telemanipulation with time delays," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 873–890, 2004.

[40] P. Huang, P. Dai, Z. Lu, and Z. Liu, "Asymmetric wave variable compensation method in dual-master-dual-slave multilateral teleoperation system," *Mechatronics*, vol. 49, pp. 1–10, 2018.

[41] L.-Y. Hu, X. P. Liu, and G.-P. Liu, "The wave-variable teleoperator with improved trajectory tracking," in *IEEE ICCA 2010*. IEEE, 2010, pp. 322–327.

[42] S. Munir and W. J. Book, "Internet-based teleoperation using wave variables with prediction," *IEEE/ASME transactions on mechatronics*, vol. 7, no. 2, pp. 124–133, 2002.

[43] B. Hannaford and J.-H. Ryu, "Time-domain passivity control of haptic interfaces," *IEEE transactions on Robotics and Automation*, vol. 18, no. 1, pp. 1–10, 2002.

[44] J.-H. Ryu, D.-S. Kwon, and B. Hannaford, "Stable teleoperation with time-domain passivity control," *IEEE Transactions on robotics and automation*, vol. 20, no. 2, pp. 365–373, 2004.

[45] Y. Ye, Y.-J. Pan, and Y. Gupta, "A power based time domain passivity control for haptic interfaces," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE, 2009, pp. 7521–7526.

[46] Z. Chen, Y.-J. Pan, J. Gu, and S. Forbrigger, "A novel multilateral teleoperation scheme with power-based time-domain passivity control," *Transactions of the Institute of Measurement and Control*, vol. 40, no. 11, pp. 3252–3262, 2018.

[47] O. J. Smith, "Closed control of loop with dead time," *Chemical engineering progress*, vol. 53, pp. 217–219, 1957.

[48] J.-Q. Huang and F. L. Lewis, "Neural-network predictive control for nonlinear dynamic systems with time-delay," *IEEE Transactions on Neural Networks*, vol. 14, no. 2, pp. 377–389, 2003.

[49] K. Fite, M. Goldfarb, and A. Rubio, "Transparent telemanipulation in the presence of time delay," in *Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003)*, vol. 1. IEEE, 2003, pp. 254–259.

[50] A. C. Smith and K. Van Hashtrudi-Zaad, "Neural network-based teleoperation using smith predictors," in *IEEE International Conference Mechatronics and Automation, 2005*, vol. 3. IEEE, 2005, pp. 1654–1659.

[51] Z. Li, Y. Xia, D. Wang, D.-H. Zhai, C.-Y. Su, and X. Zhao, "Neural network-based control of networked trilateral teleoperation with geometrically unknown constraints," *IEEE transactions on cybernetics*, vol. 46, no. 5, pp. 1051–1064, 2015.

[52] Y. Ji, D. Liu, and Y. Guo, "Adaptive neural network based position tracking control for dual-master/single-slave teleoperation system under communication constant time delays," *ISA transactions*, vol. 93, pp. 80–92, 2019.

[53] C. Secchi, A. Franchi, H. H. Bülthoff, and P. R. Giordano, "Bilateral teleoperation of a group of uavs with communication delays and switching topology," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 4307–4314.

[54] V. Duindam and S. Stramigioli, "Port-based asymptotic curve tracking for mechanical systems," *European Journal of Control*, vol. 10, no. 5, pp. 411–420, 2004.

[55] A. Franchi, C. Secchi, H. I. Son, H. H. Bulthoff, and P. R. Giordano, "Bilateral teleoperation of groups of mobile robots with time-varying topology," *Ieee transactions on robotics*, vol. 28, no. 5, pp. 1019–1033, 2012.

[56] F. Ferraguti, C. Secchi, and C. Fantuzzi, "A tank-based approach to impedance control with variable stiffness," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 4948–4953.

[57] D. Lee and K. Huang, "Passive-set-position-modulation framework for interactive robotic systems," *IEEE Transactions on Robotics*, vol. 26, no. 2, pp. 354–369, 2010.

[58] F. Ferraguti, N. Preda, M. Bonfe, and C. Secchi, "Bilateral teleoperation of a dual arms surgical robot with passive virtual fixtures generation," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 4223–4228.

[59] "G-coder systems ab," `http://g-coder.com`.

[60] "3d systems," `https://www.3dsystems.com`.

[61] W. R. Chitwood Jr, L. W. Nifong, W. H. Chapman, J. E. Felger, B. M. Bailey, T. Ballint, K. G. Mendleson, V. B. Kim, J. A. Young, and R. A. Albrecht, "Robotic surgical training in an academic institution," *Annals of surgery*, vol. 234, no. 4, p. 475, 2001.

[62] H. W. Schreuder, R. Wolswijk, R. P. Zweemer, M. P. Schijven, and R. H. Verheijen, "Training and learning robotic surgery, time for a more structured approach: a systematic review," *BJOG: An International Journal of Obstetrics & Gynaecology*, vol. 119, no. 2, pp. 137–149, 2012.

[63] B. Chebbi, D. Lazaroff, and P. X. Liu, "A collaborative virtual haptic environment for surgical training and tele-mentoring," *International Journal of Robotics & Automation*, vol. 22, no. 1, p. 69, 2007.

[64] S. S. Nudehi, R. Mukherjee, and M. Ghodoussi, "A shared-control approach to haptic interface design for minimally invasive telesurgical training," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 588–592, 2005.

[65] B. Khademian and K. Hashtrudi-Zaad, "Shared control architectures for haptic training: Performance and coupled stability analysis," *The International Journal of Robotics Research*, vol. 30, no. 13, pp. 1627–1642, 2011.

[66] S. Sirouspour and P. Setoodeh, "Multi-operator/multi-robot teleoperation: an adaptive nonlinear control approach," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 1576–1581.

[67] B. Khademian and K. Hashtrudi-Zaad, "Unconditional stability analysis of dual-user teleoperation systems," in *2010 IEEE Haptics Symposium*. IEEE, 2010, pp. 161–166.

[68] A. Ghorbanian, S. Rezaei, A. Khoogar, M. Zareinejad, and K. Baghestan, "A novel control framework for nonlinear time-delayed dual-master/single-slave teleoperation," *ISA transactions*, vol. 52, no. 2, pp. 268–277, 2013.

[69] W. Gueaieb, F. Karray, and S. Al-Sharhan, "A robust hybrid intelligent position/force control scheme for cooperative manipulators," *IEEE/ASME Transactions on Mechatronics*, vol. 12, no. 2, pp. 109–125, 2007.

[70] H. Van Quang and J.-H. Ryu, "Stable multilateral teleoperation with time domain passivity approach," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 5890–5895.

[71] Y. Wang, F. Sun, H. Liu, and Z. Li, "Passive four-channel multilateral shared control architecture in teleoperation," in *9th IEEE International Conference on Cognitive Informatics (ICCI'10)*. IEEE, 2010, pp. 851–858.

[72] L. Sabattini, C. Secchi, B. Capelli, and C. Fantuzzi, "Passivity preserving force scaling for enhanced teleoperation of multi-robot systems," *Robotics and Automation Letters*, vol. 3, no. 3, pp. 1925–1932, 2018.

[73] J. Artigas, J.-H. Ryu, and C. Preusche, "Time domain passivity control for position-position teleoperation architectures," *Presence: Teleoperators and Virtual Environments*, vol. 19, no. 5, pp. 482–497, 2010.

[74] C. Secchi, S. Stramigioli, and C. Fantuzzi, "Position drift compensation in port-hamiltonian based telemanipulation," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 4211–4216.

[75] S. Stramigioli, C. Secchi, A. J. van der Schaft, and C. Fantuzzi, "Sampled data systems passivity and discrete port-hamiltonian systems," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 574–587, Aug 2005.

[76] E. Sartori, C. Tadiello, C. Secchi, and R. Muradore, "Tele-echography using a two-layer teleoperation algorithm with energy scaling," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 1569–1575.

[77] A. Krupa, J. Gangloff, C. Doignon, M. F. De Mathelin, G. Morel, J. Leroy, L. Soler, and J. Marescaux, "Autonomous 3-d positioning of surgical instruments in robotized laparoscopic surgery using visual servoing," *IEEE transactions on robotics and automation*, vol. 19, no. 5, pp. 842–853, 2003.

[78] F. Richter, S. Shen, F. Liu, J. Huang, E. K. Funk, R. K. Orosco, and M. C. Yip, "Autonomous robotic suction to clear the surgical field for hemostasis using image-based blood flow detection," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1383–1390, 2021.

[79] H. Kang and J. T. Wen, "Autonomous suturing using minimally invasive surgical robots," in *Proceedings of the 2000. IEEE International Conference on Control Applications. Conference Proceedings (Cat. No. 00CH37162)*. IEEE, 2000, pp. 742–747.

[80] J. Schulman, A. Gupta, S. Venkatesan, M. Tayson-Frederick, and P. Abbeel, "A case study of trajectory transfer through non-rigid registration for a simplified suturing scenario," in *IROS*, 2013.

[81] F. Nageotte, P. Zanne, C. Doignon, and M. deMathelin, "Stitching planning in laparoscopic surgery: Towards robot-assisted suturing," *International Journal of Robotics Research*, vol. 28, no. 10, pp. 1303–1321, 2009.

[82] I. Elgezua, Y. Kobayashi, and M. G. Fujie, "Survey on current state-of-the-art in needle insertion robots: Open challenges for application in real surgery," *Procedia CIRP*, vol. 5, pp. 94–99, 2013.

[83] N. Preda, F. Ferraguti, G. De Rossi, C. Secchi, R. Muradore, P. Fiorini, and M. Bonfè, "A cognitive robot control architecture for autonomous execution of surgical tasks," *Journal of Medical Robotics Research*, vol. 1, no. 04, 2016.

[84] N. Preda, A. Manurung, O. Lambercy, R. Gassert, and M. Bonfè, "Motion planning for a multi-arm surgical robot using both sampling-based algorithms and motion primitives," in *IROS*, 2015.

[85] M. Cefalo, E. Magrini, and G. Oriolo, "Sensor-based task-constrained motion planning using model predictive control," in *SYROCO*, 2018.

[86] S. M. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Autonomous Robots*, vol. 32, no. 4, pp. 433–454, 2012.

[87] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization." in *Robotics: science and systems*, vol. 9, no. 1. Citeseer, 2013, pp. 1–10.

[88] B. Kehoe, G. Kahn, J. Mahler, J. Kim, A. Lee, A. Lee, K. Nakagawa, S. Patil, W. D. Boyd, P. Abbeel, and K. Goldberg, "Autonomous multilateral debridement with the raven surgical robot," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 1432–1439.

[89] V. J. Lumelsky, "On fast computation of distance between line segments," *Information Processing Letters*, vol. 21, pp. 55–61, 1985.

[90] A. Sozzi, M. Bonfè, S. Farsoni, G. De Rossi, and R. Muradore, "Dynamic motion planning for autonomous assistive surgical robots," *Electronics*, vol. 8, p. 957, 2019.

[91] N. M. de Oliveira and L. T. Biegler, "Constraint handing and stability properties of model-predictive control," *AIChE journal*, vol. 40, no. 7, pp. 1138–1155, 1994.

[92] S. Balasubramanian, A. Melendez-Calderon, and E. Burdet, "A robust and sensitive metric for quantifying movement smoothness," *IEEE transactions on biomedical engineering*, vol. 59, no. 8, pp. 2126–2136, 2011.

[93] D. S. Morris, J. T. Wei, D. A. Taub, R. L. Dunn, J. S. Wolf, and B. K. Hollenbeck, "Temporal trends in the use of percutaneous nephrolithotomy," vol. 175, pp. 1731–1736, 2006.

[94] J. de la Rosette, D. Assimos, M. Desai, J. Gutierrez, J. Lingeman, R. Scarpa, and A. Tefekli, "The clinical research office of the endourological society percutaneous nephrolithotomy global study: Indications, complications, and outcomes in 5803 patients," *Journal of Endourology*, vol. 25, no. 1, pp. 11–17, 2011.

[95] P. L. Rodrigues, N. F. Rodrigues, J. Fonseca, E. Lima, and J. Vilaca, "Kidney targeting and puncturing during percutaneous nephrolithotomy: recent advances and future perspective," *Journal of Endourology*, vol. 27, no. 7, pp. 826–834, 2013.

[96] S. A. Ziaee, M. M. Sichani, A. H. Kashi, and M. Samzadeh, "Evaluation of the learning curve for percutaneous nephrolithotomy," vol. 7, no. 4, pp. 226–231, 2010.

[97] M. Borofsky, M. Rivera, C. Dauw, A. Krambeck, and J. Lingeman, "Electromagnetic guided percutaneous renal access outcomes among surgeons and trainees of different experience levels: A pilot study," *Urology*, 2019.

[98] W. L. Strohmaier and A. Giese, "Ex vivo training model for percutaneous renal surgery," *Urological Research*, vol. 33, pp. 191–193, 2005.

[99] B. E. Knudsen, E. D. Matsumoto, B. Chew, B. Johnson, V. Margulis, J. A. Cadeddu, M. S. Pearle, S. E. Pautler, and J. D. Denstedt, "A randomized, controlled, prospective study validating the acquisition of percutaneous renal collecting system access skills using a computer based hybrid virtual reality surgical simulator: phase i," vol. 176, pp. 2173–2178, 2006.

[100] S. Mishra, A. Kurien, R. Patel, P. Patil, A. Ganpule, V. Muthu, R. B. Sabnis, and M. Desai, "Validation of virtual reality simulation for percutaneous renal access training," *Journal of Endourology*, vol. 24, no. 4, pp. 635–640, 2010.

[101] A. G. Papatsoris, T. Shaikh, D. Patel, A. Bourdoumis, C. Bach, N. Buchholz, J. Masood, and I. Junaid, "Use of a virtual reality simulator to improve percutaneous renal access skills: A prospective study in urology trainees," *Urologia Internationalis*, vol. 89, pp. 185–190, 2012.

[102] B. Sainsbury, M. Lacki, M. Shahait, M. Goldenberg, A. Baghdadi, L. Cavuoto, J. Ren, M. Green, J. Lee, T. D. Averch, and C. Rossa, "Evaluation of a virtual reality percutaneous nephrolithotomy (pcnl) surgical simulator," *Frontiers in Robotics and AI*, vol. 6, p. 145, 2020.

[103] D. Pinzon, S. Byrns, and B. Zheng, "Prevailing trends in haptic feedback simulation for minimally invasive surgery," *Surgical Innovation*, vol. 23, no. 4, pp. 415–421, 2016.

[104] J. J. Rassweiler, M. Muller, M. Fangerau, J. Klein, A. S. Goezen, P. Pereira, H.-P. Meinzer, and D. Teberd, "ipad-assisted percutaneous access to the kidney using marker-based navigation: Initial clinical experience," *European Urology*, vol. 61, no. 3, pp. 628–631, 2012.

[105] M. Muller, M. C. Rassweiler, J. Klein, A. Seitel, M. Gondan, M. Baumhauer, D. Teber, J. J. Rassweiler, H. P. Meinzer, and L. Maier-Hein, "Mobile augmented reality for computer-assisted percutaneous nephrolithotomy," *International Journal of Computer Assisted Radiology and Surgery*, vol. 8, pp. 663–675, 2013.

[106] H. Li, I. Paranawithana, Z. H. Chau, L. Yang, T. S. K. Lim, S. Foong, F. C. Ng, and U. Tan, "Towards to a robotic assisted system for percutaneous nephrolithotomy," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, 2018.

[107] P. Vavra, J. Roman, P. Zonca, P. Ihnat, M. Nemec, J. Kumar, N. Habib, and A. El-Gendi, "Recent development of augmented reality in surgery: A review," *Journal of healthcare engineering*, vol. 2017, 2017.

[108] L.-M. Su, D. Stoianovici, T. Jarrett, A. Patriciu, W. Roberts, J. Cadeddu, S. Ramakumar, S. Solomon, and L. Kavoussi, "Robotic percutaneous access to the kidney: Comparison with standard manual access," *Journal of endourology / Endourological Society*, vol. 16, pp. 471–4755, 10 2002.

[109] P. Cinquin, "How today's robots work and perspectives for the future," *Journal of visceral surgery*, vol. 148, no. 5S, pp. e12–e18.

[110] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image and vision Computing*, vol. 76, pp. 38–47, 2018.

[111] P. Corke, *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised*. Springer, 2017, vol. 118.

[112] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," in *Advances in psychology*. Elsevier, 1988, vol. 52, pp. 139–183.

[113] M. M. Dalvand and B. Shirinzadeh, "Motion control analysis of a parallel robot assisted minimally invasive surgery/microsurgery system (pramiss)," *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 2, pp. 318–327, 2013.

[114] M. C. Çavuşoğlu, W. Williams, F. Tendick, and S. S. Sastry, "Robotics for telesurgery: Second generation berkeley/ucsf laparoscopic telesurgical workstation and looking towards the future applications," *Industrial Robot: An International Journal*, 2003.

[115] H. Rininsland, "Artemis. a telemanipulator for cardiac surgery," *European Journal of Cardio-Thoracic Surgery*, vol. 16, no. Supplement_2, pp. S106–S111, 1999.

[116] B. Eldridge, K. Gruben, D. LaRose, J. Funda, S. Gomory, J. Karidis, G. McVicker, R. Taylor, and J. Anderson, "A remote center of motion robotic arm for computer assisted surgery," *Robotica*, vol. 14, no. 1, pp. 103–109, 1996.

[117] Y. Wang, D. Uecker, K. P. Laby, J. D. Wilson, C. S. Jordan, J. W. Wright, and M. Ghodoussi, "Medical robotic arm that is attached to an operating table," Aug. 1 2006, uS Patent 7,083,571.

[118] B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial," *Journal of intelligent and robotic systems*, vol. 3, no. 3, pp. 201–212, 1990.

[119] U. Hagn, R. Konietschke, A. Tobergte, M. Nickl, S. Jörg, B. Kübler, G. Passig, M. Gröger, F. Fröhlich, U. Seibold, *et al.*, "Dlr mirosurge: a versatile system for research in endoscopic telesurgery," *International journal of computer assisted radiology and surgery*, vol. 5, no. 2, pp. 183–193, 2010.

[120] T. Osa, C. Staub, and A. Knoll, "Framework of automatic robot surgery system using visual servoing," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 1837–1842.

[121] H. Azimian, R. V. Patel, and M. D. Naish, "On constrained manipulation in robotics-assisted minimally invasive surgery," in *2010 3rd IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics*. IEEE, 2010, pp. 650–655.

[122] B. Siciliano and J.-J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Fifth International Conference on Advanced Robotics' Robots in Unstructured Environments*. IEEE, 1991, pp. 1211–1216.

[123] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-priority based redundancy control of robot manipulators," *The International Journal of Robotics Research*, vol. 6, no. 2, pp. 3–15, 1987.

[124] B. Dahroug, B. Tamadazte, and N. Andreff, "Task controller for performing remote centre of motion," in *Informatics in Control, Automation and Robotics*. Springer, 2018, pp. 117–132.

[125] M. Michelin, P. Poignet, and E. Dombre, "Dynamic task/posture decoupling for minimally invasive surgery motions: simulation results," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 4. IEEE, 2004, pp. 3625–3630.

[126] J. Sandoval, G. Poisson, and P. Vieyres, "A new kinematic formulation of the rcm constraint for redundant torque-controlled robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 4576–4581.

[127] H. Sadeghian, F. Zokaei, and S. H. Jazi, "Constrained kinematic control in minimally invasive robotic surgery subject to remote center of motion constraint," *Journal of Intelligent & Robotic Systems*, vol. 95, no. 3-4, pp. 901–913, 2019.

[128] J. Funda, R. H. Taylor, B. Eldridge, S. Gomory, and K. G. Gruben, "Constrained cartesian motion control for teleoperated surgical robots," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 3, pp. 453–465, 1996.

[129] R. C. Locke and R. V. Patel, "Optimal remote center-of-motion location for robotics-assisted minimally-invasive surgery," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 1900–1905.

[130] A. Albu-Schaffer and G. Hirzinger, "Cartesian impedance control techniques for torque controlled light-weight robots," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 1, 2002, pp. 657–663 vol.1.

[131] M. Maier, K. Kondak, and C. Ott, "Enabling robot assisted landing of heavy uav rotorcraft via combined control and workload sharing," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 5128–5134.

[132] A. Casals, A. Hernansanz, N. Sayols, and J. Amat, "Assistance strategies for robotized laparoscopy," in *Robot 2019: Fourth Iberian Robotics Conference*, Springer, Ed., 2019, pp. 485–496.

[133] F. E. Udwadia and R. E. Kalaba, "A new perspective on constrained motion," *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, vol. 439, no. 1906, pp. 407–410, 1992.

[134] J. Huang, Y.-H. Chen, and Z. Zhong, "Udwadia-kalaba approach for parallel manipulator dynamics," *Journal of Dynamic Systems, Measurement, and Control*, vol. 135, no. 6, 2013.