POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Biomedica
Dipartimento di Elettronica, Informazione e Bioingegneria

INTRAOPERATIVE SAFETY CONSTRAINTS DEFINITION SYSTEM FOR ROBOTIC MINIMALLY INVASIVE SURGERY

Laboratorio di Neuroingegneria e Robotica Medica

Relatore:
Prof. Elena De Momi
Correlatori:
Dr. Thibaud Chupin, Dr. Andrea Mariani

Tesi di Laurea di
Luca Vantadori, Matr. 853314

Anno Accademico 2017-2018

Contents

List of Figures

List of Tables

Sommario

Il tema di ricerca del seguente lavoro è guidato da una forte necessità clinica di nuove ed efficienti tecnologie per aiutare la figura del chirurgo moderno. Le tecniche chirurgiche si stanno evolvendo rapidamente e, grazie all'avvento della chirurgia robotica mininvasiva, gli esiti per i pazienti sono in continuo miglioramento, caratterizzati da recuperi più rapidi, dolori postoperatori ridotti e risultati estetici migliori.

Tuttavia, l'esito della chirurgia mininvasiva, anche quando eseguita con un sistema robotico, è influenzato dal sanguinamento intraoperatorio e dalle disfunzioni postoperatorie. La causa primaria è attribuita al danno accidentale a vasi e nervi. L'esito della procedura chirurgica dipende quindi interamente dall'abilità e dalla destrezza del chirurgo.

Per tale motivo vi è una crescente necessità di sviluppare soluzioni ingegneristiche innovative per aiutare i chirurghi a raggiungere il miglior risultato clinico possibile.

Questa tesi presenta un "Sistema di Definizione dei Vincoli di Sicurezza Intraoperatori per la Chirurgia Robotica Mininvasiva" che offre al chirurgo la possibilità di definire un'area di sicurezza intraoperatoria. L'informazione 3D, racchiusa all'interno dell'area, viene sfruttata per ricostruire la superficie di un volume di sicurezza, ovvero una regione di spazio in cui lo strumento robotico non deve entrare. La realtà aumentata viene utilizzata per visualizzare il volume di sicurezza proiettato sull'immagine e la distanza tra il tessuto e lo strumento chirurgico, mentre un feedback di forza allontana quest'ultimo dalla struttura critica selezionata.

Il sistema è stato integrato nel da Vinci Research Kit (dVRK, fornito da Intuitive Surgical al Politecnico di Milano) che comprende cinque manipolatori (due master, due slave, un braccio per la camera), un endoscopio e una console di visione stereoscopica.

Per convalidare l'utilità del sistema, è stato sviluppato un ambiente virtuale dove poter condurre gli esperimenti. È stata ricreata una simulazione

di nefrectomia parziale con modelli anatomicamente corretti di tumore renale e arterie renali, entrambi posizionati su una superficie rappresentante il rene.

A dieci candidati volontari è stato chiesto di eseguire la rimozione del tumore cercando di evitare la collisione dello strumento robotico con le arterie renali. Ogni candidato ha eseguito la prova in due modalità diverse: nella prima la simulazione prevedeva la semplice rimozione del tumore senza l'aiuto di alcun feedback visivo o tattile; nella seconda il candidato doveva definire un'area di sicurezza in modo che l'esecuzione della prova fosse supportata dal feedback di forza generato dalla superficie del volume di sicurezza e dagli aiuti visivi che indicano la distanza dello strumento dal tessuto.

Per valutare l'utilità e l'efficienza del sistema sono stati analizzati il numero di collisioni dello strumento robotico con le arterie renali, la durata totale delle collisioni e la percentuale di tessuto sano rimosso in aggiunta al tumore per ciascun candidato e ciascuna prova.

I risultati hanno sottolineato come il sistema concepito abbia aumentato l'accuratezza dell'esecuzione della prova di rimozione del tumore. Infatti, il numero di collisioni, la durata totale di quest'ultime e la percentuale di tessuto sano rimosso oltre al tumore sono risultati inferiori nella prova svolta in presenza dei vincoli di sicurezza. Inoltre, i tempi di rimozione del tumore delle prove hanno mostrato come i candidati abbiano completato l'esperimento più velocemente grazie ai vincoli, evidenziando una riduzione sul carico cognitivo.

I risultati ottenuti sono incoraggianti e permettono di supporre che il sistema concepito abbia migliorato i metodi preesistenti, superando i limiti che li caratterizzano.

Il lavoro corrente è parte del progetto "SMARTsurg" (SMart weArable Robotic Teleoperated surgery), il quale ha ricevuto i fondi dall' "European Union's Horizon 2020 research and innovation programme". Nella piattaforma proposta verranno integrate soluzioni avanzate tra cui dei guanti esoscheletrici indossabili per controllare gli strumenti chirurgici e un sistema per la definizione dei vincoli di sicurezza intraoperativi come mostrato in questo lavoro [59].

Per concludere, il lavoro presentato in questa tesi potrebbe rappresenta-

re non solo un valido strumento per la chirurgia robotica, ma un punto di partenza per le future ricerche nel campo delle nuove tecnologie chirurgiche.

Abstract

The research theme of this work is driven by a strong clinical need for new and improved technologies to help the modern surgeon. Surgical techniques are evolving rapidly and, thanks to the advent of minimally invasive surgery and robot-assisted surgery, the outcomes for patients are continually improving with faster recovery, reduced postoperative pain, and better cosmetic results.

However, the outcome of minimally invasive surgery, even when performed with a robotic system, is affected by intraoperative bleeding and postoperative dysfunctions. The primary cause is the accidental damage to vessels and nerves. The outcome of the surgical procedure falls then entirely on the surgeon's skill, dexterity, and experience.

For such a reason, there is an increasing need to develop innovative engineering solutions to help surgeons to achieve the best possible clinical outcome.

This thesis presents an "Intraoperative Safety Constraints Definition System for Robotic Minimally Invasive Surgery" which gives the surgeon the possibility to define an intraoperative safety area. The 3D information, enclosed inside the area, is exploited to reconstruct the surface of a safety volume, a region of space in which the robotic tool should not enter. Augmented reality is used to visualize the safety volume projected on the image and to display the distance between the tissue and the instrument tip, while force feedback steers the robotic tool away from the selected critical structure. The system was integrated on the da Vinci Research Kit (dVRK, provided by Intuitive Surgical to Politecnico di Milano) that comprises five manipulators (two masters, two slaves, one camera arm), a stereo endoscope and a stereo vision console.

To validate the system usability under realistic conditions, a virtual environment was developed for the experimentation. A virtual partial nephrectomy simulation was created with anatomically correct models of a renal tumor and renal arteries, both placed on a stand-in for the kidney.

Ten volunteer candidates were asked to perform the removal of the tumor whilst trying to avoid the collision of the robotic tool with the renal arteries. Each candidate performed the task in two different modalities: in the first unconstrained modality, the simulation provided for the simple removal of the tumor without the help of any visual or tactile feedback; in the second constrained modality, the candidate had to define a safety area so that the execution of the task was supported by the force feedback generated from the surface of the safety volume, as well as visual aids indicating the distance between the tool and the tissue.

To evaluate the utility and efficiency of the system, the number of collisions of the robotic tool with the renal arteries, the total duration of the collisions and the percentage of healthy tissue removed in addition to the tumor were analyzed for each candidate and each modality.

The results highlighted how the system conceived increased the accuracy of completing the tumor removal task. In fact, the number of collisions, the total duration of the latter and the percentage of healthy tissue removed in addition to the tumor were lower in the constrained test. Also, the tumor removal times showed how the candidates completed the task faster thanks to the safety constraints, highlighting a reduction of the cognitive load.

The results obtained are encouraging and allow us to infer that the system conceived has improved the pre-existing methods, exceeding the limits that characterize them.

The current work is part of the "SMARTsurg" (SMart weArable Robotic Teleoperated surgery) project, which has received funding from the "European Union's Horizon 2020 research and innovation programme". Advanced features will be integrated into the proposed platform including dexterous anthropomorphic wearable hand exoskeletons with haptic feedback to control the surgical instruments and a system for defining intraoperative safety constraints as shown in this work [59].

To conclude, the work presented in this thesis could represent not only a valid intraoperative tool for robotic surgery but also a starting point for future research in the field of new surgical technologies.

Chapter 1

Introduction

1.1 Background: research motivation

Most of the studies and progress in surgery have mainly focused on minimizing the invasiveness of surgical procedures. Consequently, today there has been a significant methodological change in different surgical procedures: surgeons do not directly see and, especially, do not directly touch the anatomical structures on which they operate. In fact, advances in video imaging, endoscopic technology, and robotic instrumentation have made possible a definite transition from traditional surgery to laparoscopic surgery. Furthermore, the use of computers and surgical robots allows improving the surgeon's precision in case of micro-scale operations and his skills in learning new complex techniques.

Robotic surgery has drastically changed specific surgical procedures, simplifying and, at the same time, improving the work of the healthcare professional. Through the use of a mechanized interface and a visual interface, this new mode allows physicians to perform increasingly complex minimally invasive operations while improving the outcome of the procedures and the patient safety.

However, being the surgical robots controlled by the healthcare professional, they are not able to entirely remove the human error. For this reason, the latest new robotic intraoperative systems can be considered an essen-

tial resource in terms of increasing the surgeon's dexterity and the patient's safety during the operation.

The review of the literature, concerning the fields of robotic surgery, allows us to conclude that robotic systems and tools can be extremely useful to improve the skills and dexterity of surgeons in the most complex procedures, the management and safety of patients, and the competence in performing the intervention. The development of these fields will depend strongly on their impact on the patient safety, ease of execution of procedures and cost efficiency [1].

1.2 Problem statement

In minimally invasive surgery, even when performed with a robotic system, the most common complication is accidental damage to nerves, veins, and arteries. This may cause dysfunctions, heavy bleeding and affect the outcome of the surgical procedure. The main risk factors, during the surgical procedure, fall entirely on the surgeon's skills [2].

An avenue of research, to mitigate this issue, is to use active constraints (also known as safety constraints) to steer the robotic tools away from fragile anatomical structures.

Preservation of vessels, arteries and other highly sensitive and fragile anatomical areas, during the intraoperative phase, was studied by registering preoperative information on the patient and visualizing them through an augmented reality system [3]. However, this method can not consider the dynamic changes of the anatomical structures that can occur in the time interval between the preoperative acquisition phase and the subsequent surgical procedure. To update the preoperative/intraoperative registration in real time, tracking and reconstruction algorithms based on stereoscopic images can be used [4].

Although these algorithms have achieved high levels of performance, they nevertheless require certain characteristics that are fundamental to the intended purpose [5]:

– high accuracy

– robustness in adverse circumstances (e.g., sudden camera movements, camera occlusion)

– adaptation to changes in the patient's anatomy

– real-time processing

Taking into account the observations listed above and being aware of research and innovation play a vital role in robotic surgery, some solutions are developed in this research.

More in details, this thesis presents an "Intraoperative Safety Constraints Definition System for Robotic Minimally Invasive Surgery" which allows the surgeon, during the intervention, to define a safety area. The 3D information is used to identify the safety volume, a region of space in which the robotic tools should not enter, fitted around the selected tissue surface. Various visual and tactile cues let the surgeon steer the robotic tools away from it. Augmented reality is used to visualize at run-time the safety volume projected on the image and to display the tissue-instrument distance. The system was integrated into the da Vinci Research Kit (dVRK, provided by Intuitive Surgical to Politecnico di Milano) to validate its usability under realistic conditions.

In particular, two of the primary surgical procedures performed with the da Vinci surgical system are introduced: the radical prostatectomy and the partial nephrectomy. These procedures are ideally suited to explain the motivations that led to the realization of the system conceived.

1.3 Contributions

The major contributions of this research are summarized below:

- a method for selecting and reconstructing safety volumes of the selected anatomical parts.

- a system of safety constraints to steer the robotic tool away from the safety volume.
- a surgical virtual environment useful to validate the system performance.

1.4 Outline

This dissertation is organized in the following manner:

- Chapter 1 introduces the limits of the current technologies for minimally invasive surgery and the importance of overcoming the mentioned limitations by devising innovative engineering solutions. As a result, the tools and the system listed above are developed.
- Chapter 2 describes the state of art of the sector previously treated and its current problems, in order to provide a general overview of the research area of the issue faced.
- Chapter 3 describes the goal of the research, the methods developed to overcome the current limits and the innovations introduced. It also shows the architecture of the whole system, explaining step by step the various modules that compose it.
- Chapter 4 shows the experimental activities carried out and the results obtained with their critical evaluation in order to determine the efficiency of the system.
- Chapter 5 presents the conclusions concerning the project, the future perspectives of the research related to the area of study and the possible improvements.

Chapter 2

Background to the research problem: literature review

2.1 Introduction

Thanks to the advent and rapid development of new technologies, it has been possible to develop systems aimed at minimizing the discomfort and traumatic hospitalization due to the surgical intervention and maximizing its therapeutic success. With these objectives, minimally invasive surgery was born and developed; first only laparoscopic, then also robot-assisted.

2.2 Laparoscopic surgery

Laparoscopic surgery (Fig. 2.1), now considerably introduced into daily surgical practice, is an alternative way of dealing with surgery. Laparoscopy reduces the surgical trauma, allowing to perform almost all the same operations of traditional surgery without practicing a laparotomy (extended incision of skin, muscle fascia and muscles) [6].

Figure 2.1: Laparoscopic surgery [1]

The main advantage of laparoscopic surgery is a far inferior surgical invasiveness compared to the classic open surgery.

In this way, the pain and incidence of infections at the surgical site decrease, as well as recovery times and hospital stays, while a better aesthetic result is obtained.

According to the medical community, the innovation and progress that this technique has brought into the world of surgery are enormous but, despite the undisputed merits, there are many issues and limitations relating to the nature of the tools used [7].

First, the operative field is seen through a 2D monitor with loss of the depth-perception. The natural hand-eye coordination is compromised since the surgeon must coordinate the eyes and the hands through the monitor: moving the laparoscopic instruments looking at a two-dimensional monitor placed on a different axis is not very intuitive. Also, fundamental is the

reduction of tactile feedback and strength that makes the manipulation of the tissues much more dependent on the visual feedback. There's also a lower precision of the movements, due to the need to use long and rigid instruments which, moreover, have a limited number of degrees of freedom (DoF): most, in fact, offers just four DoF instead of seven like a human wrist. This makes common tasks, such as suturing, awkward to perform. In laparoscopic interventions, there is also a reduced dexterity in movements due to the fulcrum effect, which consists in the fact that the surgeon, to correctly interact with a structure of interest of the patient, must move the instruments in the opposite direction to the target on the monitor. Finally, the physiological tremor of the human hand is transmitted to the laparoscopic instruments. All these factors make the dissections extremely delicate, and the risk of bleeding and damage to vessels and organs remains however high and harder to correct [8].

2.3 Robot-assisted minimally invasive surgery

Robotic surgery was born to overcome the limitations of laparoscopy and to expand the benefits of minimally invasive surgery. Surgeons can operate through the same small accesses used in laparoscopy but with more precision and a broader range of movements than laparoscopic surgery (Fig. 2.2).

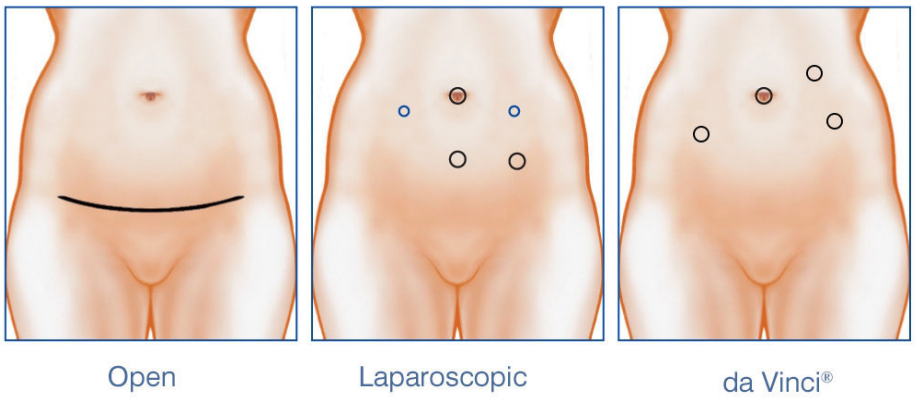


Figure 2.2: Example of incisions for different surgical approaches [2]

Therefore, the robotic approach allows combining the advantages of minimally invasive surgery with a more comfortable and better technical implementation of the procedures [9].

The goal of robotic surgery is to create a completely integrated system that converts information into action. Through information gathering and networking, navigation and guidance, dexterity enhancement and simulation of virtual environments, it is possible to transcend human limitations on a microscale basis or areas of the body difficult to access. By placing a computer between the surgeon's hand and the tip of the surgical instrument, microscale (superhuman) tasks become achievable. The principal improvements of these systems are motion scaling, tremor filtering, and no fulcrum effect returning camera control to the surgeon [10].

Robot-assisted mininvasive surgery (RAMIS) is very promising to improve the accuracy and dexterity of a surgeon while minimizing patient trauma. However, the widespread clinical success with RAMIS is still quite marginal, and it is hypothesized that the lack of tactile feedback presented to the surgeon is a limiting factor. In minimally invasive robot-assisted surgery, all natural tactile feedback is eliminated because the surgeon no longer directly manipulates the instruments. So, the risk of damaging the patient's anatomical areas and causing bleeding during the surgery can remain high [11].

2.3.1 The da Vinci Surgical System

The use of robotic surgical technique begins with applications in the neurosurgical field, thanks to the creation of PUMA 560, used in 1985 to perform neurosurgical biopsies under CT guidance with higher precision compared to conventional methods. Subsequently, in 1988, it was the turn of PROBOT, a robot created specifically to perform transurethral prostatic resections. While PROBOT was being developed, the Integrated Surgical Supplies in Sacramento, California, created in 1992 ROBODOC, a robotic system designed to be used in hip prosthetic surgery. ROBODOC was the first robot to receive the FDA approval for use in the medical field. In 1994 Com-

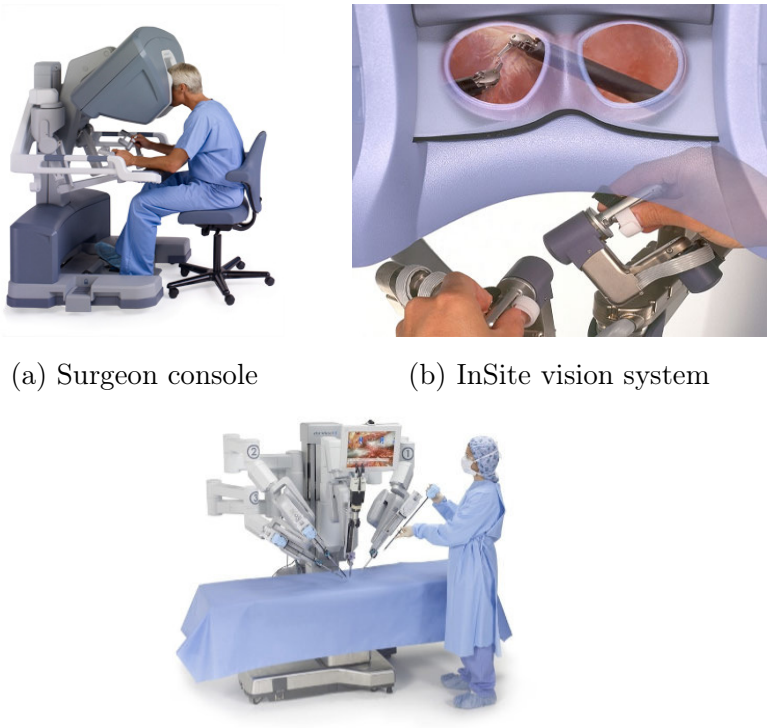
puter Motion in California created AESOP (Automated Endoscopy System for Optimal Positioning), a robotic arm controlled by the surgeon using voice commands that can manipulate a laparoscopic camera. Shortly afterward, Computer Motion put into production ZEUS, a robot composed of a console and a slave unit, the latter formed by two robotic arms and a vocal control endoscopic arm. It was with this machine that in September 2001 the first remote distance intervention was carried out between New York and Strasbourg. However, the real innovation happened with the creation of the Robotic da Vinci System (Fig. 2.3) by Intuitive Surgical, California. Behind the creation of this robot, one of the main driving forces is the interest in developing telesurgery, that is a remote surgery employing robotic instruments [12].

Figure 2.3: The da Vinci Surgical System [3]

The da Vinci surgical system is comprised of three components:

- the first component is the surgeon console, where the surgeon sits away from the patient and uses a stereoscopic viewer with hand manipulators and foot pedals that allow controlling the robotic instruments within the patient's body (Fig. 2.4a).

- the second component is the InSite vision system, which provides the three-dimensional image through a 12-mm endoscope containing stereoscopic cameras and dual optical lenses (Fig. 2.4b).
- the third component is the patient-side cart with telerobotic arms and Endowrist instruments. Currently, this system is available with either three or four robotic arms. One of the arms holds the laparoscope while the other two or three arms hold the various laparoscopic surgical instruments (Fig. 2.4c).



(a) Surgeon console

(b) InSite vision system

(c) Patient-side cart

Figure 2.4: da Vinci Surgical System components [4]

The da Vinci Surgical System is, therefore, a master/slave robot, where the surgeon is the master who controls every action of the slave robot and the console represents the necessary interface between the two. An infrared head sensor freezes the robotic arms whenever the surgeons removes his eyes from

the vision device. The console dome is designed to obstruct the surgeon's peripheral vision as a sort of blinders so that when he inserts the head into the vision device, he is fully immersed in the 3D operating field as if he were inside the patient. This type of perception helps in maintaining anatomical orientation. The dome of the console is therefore designed to respond to the need to isolate the surgeon from the surrounding environment; a need particularly felt if the da Vinci is used for telesurgery as was initially assumed.

The surgeon's arms are placed on a special padded shelf, and his fingers grab the master controllers that convert the three-dimensional movements of the surgeon's hands into electrical signals, which are then translated by the computer into commands that direct the robotic instruments. In this way, they perform identical but scaled down three-dimensional movements on the patient.

It should be kept in mind that the master controllers also provide a minimum tactile and force feedback. For example, while the surgeon is performing a suture, they indicate through the resistance to movement the tension applied to the suture material by the two active robotic arms. In practice, however, the surgeon mainly receives tactile and forceful information through visual cues.

At the base of the console, there is a pedal board through which the surgeon can control other functions of the robot. In detail, considering the da Vinci Research Kit used in this work, the pedal on the far right activates the arms, the medial on the right is unused, the central pedal modifies the focus of the cameras, the medial on the left allows to control the camera arm and the pedal on the far left finally blocks the robotic instruments in a static position: the surgeon can release the master controllers from the robotic instruments, which remain motionless in the last position acquired, so that the masters can be positioned in a more comfortable pose.

The da Vinci patient-side cart is placed next to the operating table at the beginning of the operation with a different location depending on the type of the surgical procedure. The system, as initially designed, consisted of only three arms while in December 2002 the FDA approved a new generation of da Vinci equipped with four arms. In both models, the central arm is designed

to support the camera that the surgeon can move through the master controllers. The laparoscopic instruments are attached to the remaining lateral robotic arms [13].

2.4 Radical prostatectomy: anatomy and approaches

The prostate is a compound tubuloalveolar exocrine gland of the male reproductive system in most mammals. It is situated in the male pelvis, below the urinary bladder and surrounds the urethra, which carries urine from the bladder to the penis. Radical prostatectomy refers to the surgical removal of the prostate gland [14].

Analysis of annual surgeon caseload, in the UK, revealed that 54% of surgeons performed an average of fewer than ten procedures per year and 6% of surgeons performed an average of 30 or more procedures per year [15].

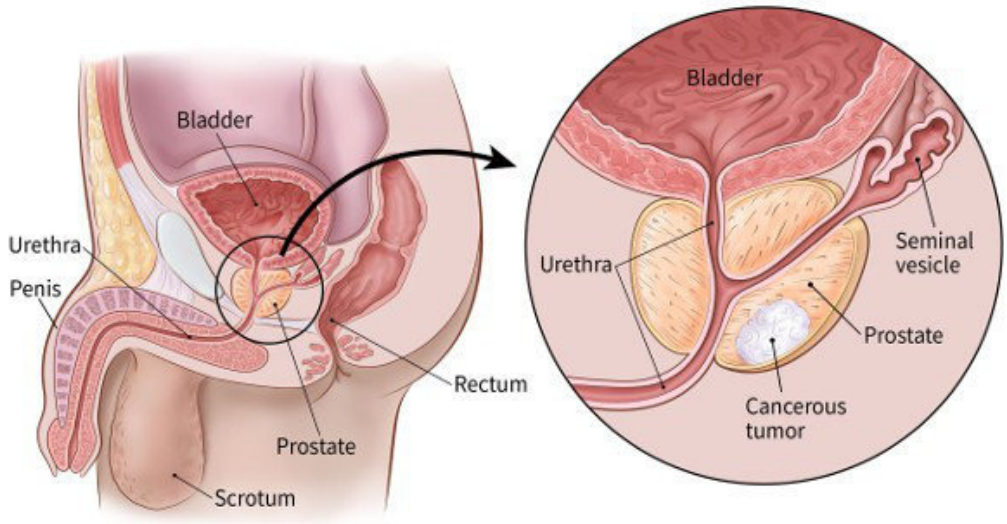


Figure 2.5: Prostate anatomy and related cancer [5]

Radical prostatectomy is done for benign conditions that cause urinary retention, recurrent urinary tract infections, and uncontrollable hematuria

(presence of red blood cells in the urine), as well as for prostate cancer and other cancers of the pelvis [16] (Fig. 2.5).

In the United States, prostate cancer accounts for approximately one-third of cancers in men. It has been estimated that the annual incidence of prostate cancer in the United States will rise from 192.280 in 2009 to 384.000 in the year 2025 and 452.000 in the year 2045 [17].

Radical prostatectomy has a low risk of serious complications. Death or serious disability caused by radical prostatectomy is extremely rare. However, important nerves travel through the prostate on the way to the penis and complications from inadvertent nerve damage do occur after radical prostatectomy [18].

They include [19] [20] [21]:

- Urinary incontinence. Incontinence involves uncontrollable, involuntary leaking of urine, which may improve over time, even up to a year after surgery. This symptom may be worse if the patient is older than age 70 when the surgery is performed.

- Urinary leakage or dribbling. This symptom is at its worst immediately after the surgery, and will usually improve over time.

- Erectile dysfunction, also known as impotence. Recovery of sexual function may take up to two years after surgery and may not be complete. Nerve-sparing prostatectomy lessens the chance of impotence but doesn't guarantee that it will not happen.

- Sterility. Cutting the connection between the testicles and the urethra causes retrograde ejaculation. This results in a man being unable to provide sperm for a biological child. A man may be able to have an orgasm, but there will be no ejaculate.

- Lymphedema. Lymphedema is a condition in which fluid accumulates in the soft tissues, resulting in swelling. Lymphedema may be caused by inflammation, obstruction, or removal of the lymph nodes during surgery. Although this complication is rare, if lymph nodes are removed

during prostatectomy, fluid may accumulate in the legs or genital region over time. Pain and swelling result.

Much of the skill involved in radical prostatectomy centers on sparing the nerves during the operation. For this reason, the type of surgery, the methods and the dexterity of the surgeon are fundamental.

Surgeons can choose from three different approaches to reach and remove the prostate during a radical prostatectomy [22]:

- Open surgery
- Laparoscopic surgery
- Robot-assisted surgery

2.4.1 Open surgery

In an open radical prostatectomy, the prostate is accessed through a large single incision through either the lower abdomen or the perineum (Fig. 2.6).

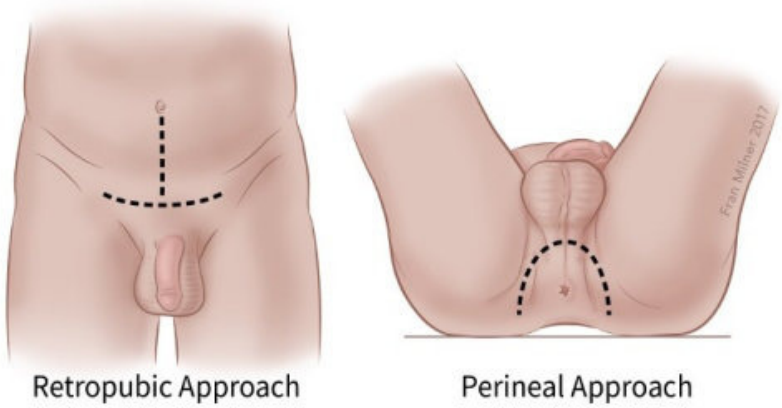


Figure 2.6: Open prostatectomy incisions [6]

Further descriptive terms describe how the prostate is accessed anatomically through these incisions:

- Retropubic approach: it describes a procedure that accesses the prostate by going through the lower abdomen and behind the pubic bone [23].
- Perineal approach: it describes a procedure that makes an incision between the rectum and scrotum on the underside of the abdomen [24].

2.4.2 Laparoscopic surgery

The main technique is the transperitoneal approach, to accommodate for sufficient working space for the trocars, and to access the seminal vesicles. Five trocars are placed in a fan array configuration (Fig. 2.7):

- a 10-mm umbilical trocar is placed for the laparoscope.
- a 10-mm trocar is positioned on the right side on the lateral edge of the rectus muscle.
- a 5-mm trocar is placed on the left lateral edge of the rectus abdominis muscle.
- two 5-mm trocars are inserted approximately 2 cm medial and superior to the anterior superior iliac spines.

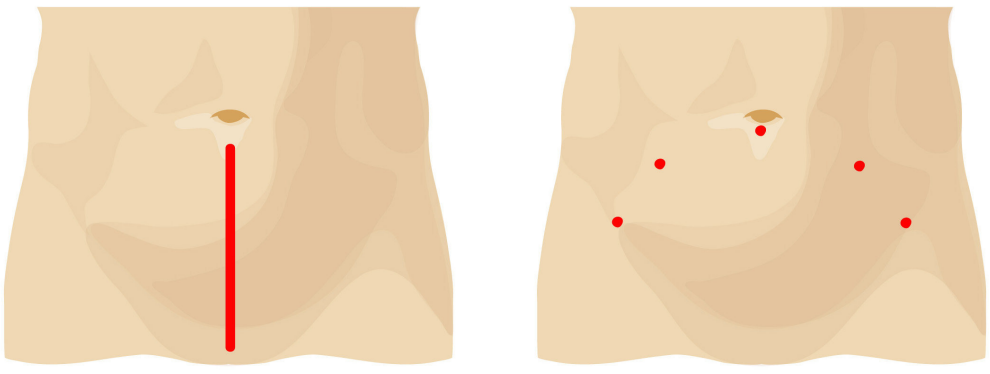


Figure 2.7: Open versus laparoscopic prostatectomy incisions [7]

Pneumoperitoneum, the presence of air in the peritoneal cavity to distends and separates the abdominal wall from its contents, is usually obtained using a Veress needle (Fig. 2.8).

Figure 2.8: Veress needle [8]

2.4.3 Robot-assisted surgery

The robotic radical prostatectomy is performed similarly to traditional laparoscopic prostatectomy, by making small incisions on the patient's abdomen. In this case, however, the surgeon does not work directly on the patient but performs the procedure using the surgical robot (e.g., da Vinci Surgical System). The miniaturized robotic instruments are passed through the several small keyhole incisions in the patient's abdomen to allow the surgeon to remove the prostate and nearby tissues, as well as the three-dimensional endoscope, used to provide a magnified view of delicate structures surrounding the prostate gland (e.g., nerves, blood vessels and muscles).

2.4.4 Comparison of approaches

There is general agreement that the goals of radical prostatectomy (RP) are, in order of importance, to cure cancer, maintain urinary continence, maintain erectile function and minimize complications [25].

For many procedures, an advantage of a laparoscopic approach, as al-

ready mentioned, is its less invasive aspect when compared with an open surgical incision. With laparoscopy, minimally invasive incisions may create less postoperative pain, decrease the analgesic requirement and reduce the hospitalization period.

Operative outcomes

Open radical prostatectomy (ORP) is performed through an 8 to 10 cm lower abdominal incision. Both laparoscopic radical prostatectomy (LRP) and robot-assisted laparoscopic radical prostatectomy (RALRP) make use of smaller incisions. For RALRP, the cumulative size of the incisions is approximately 5 to 6 cm [26].

Other advantages of the laparoscopic approach to prostatectomy include improved visualization and the positive pressure created by the carbon dioxide pneumoperitoneum used for insufflation. Pneumoperitoneum reduces the pressure gradient between the blood vessels and the remainder of the operative field, resulting in less venous and capillary bleeding during the operation. It has been showed that blood loss is consistently reduced in the LRP and RALRP approaches [27].

An important factor supporting the robotic technique is the rate of conversion to open surgery. When surgeons performing LRP and RALRP encounter serious complications, they may need to convert emergently to an open procedure to control life-threatening bleeding. Most studies report conversion rates for LRP of 2% to 8%, compared with 0% to 1% for RALRP [28].

Oncological outcomes

The primary goal of prostate cancer surgery is to provide satisfactory oncological outcomes. Biochemical progression is one of the commonly used indices to assess oncological outcomes following RP. A rising prostate-specific antigen (PSA) level is an early sign of biochemical progression and prostate cancer recurrence. Overall, different studies report that the 5-year freedom from PSA failure rates is higher, in average, for RALRP, followed by ORP

and then LRP (90%, 80% and 75% respectively) [29].

Functional Outcomes

Urinary incontinence and erectile dysfunction are the two major functional concerns for patients after radical prostatectomy.

Continence rate, commonly defined as requiring one or no pads per day, is reported to be between 90% to 92% after ORP, 82% to 96% after LRP, and 95% to 96% after RALRP [29].

It is difficult based on the current literature to determine if one approach is superior to the other for the preservation of the neurovascular bundles and sexual function. However, using two studies included in the meta-analysis by Parsons and Bennett, there was a trend toward increased potency for the laparoscopic and robotic-assisted group. Among the patients who underwent a bilateral nerve-sparing procedure, 71% of LRP patients and 76.5% of RALRP patients were potent at 12 months postoperatively [30].

2.5 Partial nephrectomy: anatomy and approaches

The kidneys are two bean-shaped organs present in left and right sides of the body. They are located at the back of the abdominal cavity. They receive blood from the paired renal arteries; blood exits into the paired renal veins. Each kidney is attached to a ureter, a tube that carries excreted urine to the bladder (Fig. 2.9). Partial nephrectomy is the surgical removal of a kidney tumor along with a thin rim of healty tissue, with the two aims of curing cancer and preserving as much healthy kidney as possible.

The incidence of kidney cancer has been steadily increasing, with an estimated 64,770 new cases diagnosed in the United States in 2012. From 2002 to 2010, 2,912 urologists performed a total of 17,640 open radical nephrectomies (ORNs), 1,558 urologists performed 7,104 open partial nephrectomies (OPNs), 2,340 urologists performed 18,852 laparoscopic radical nephrectomies (LRNs), and 853 urologists performed 4,788 laparoscopic partial

nephrectomies (LPNs). The annual proportion of ORN decreased by half, from 54% of all nephrectomies in 2003 to 29% in 2010. At the same time, it can be observed an increase in the annual proportion of LPN, from 2% in 2003 to 17% in 2010 [31].

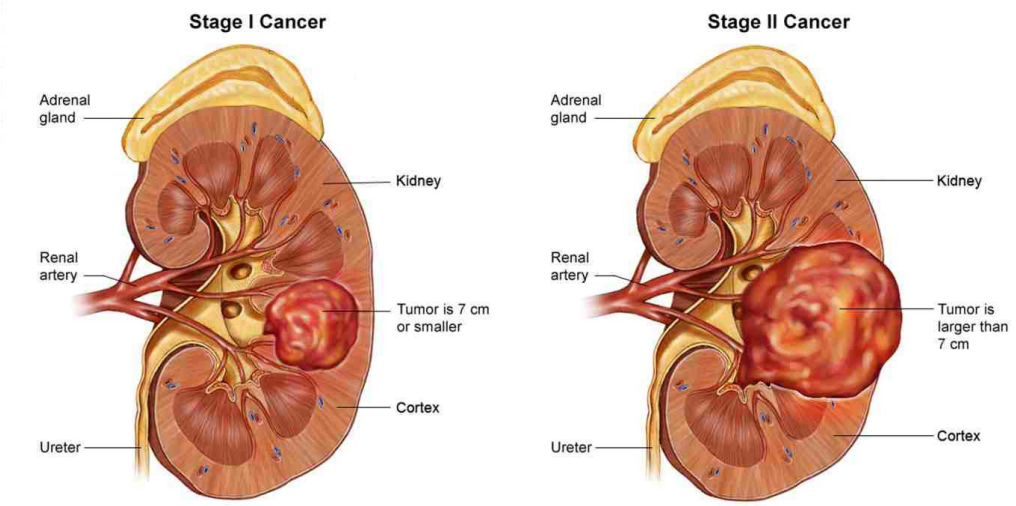


Figure 2.9: Kidney anatomy and related cancer [9]

Therefore, during the past two decades, two significant treatment advances have been made. The first was the expanded use of the partial nephrectomy. The other major treatment change came with the minimally invasive era. The first laparoscopic partial nephrectomy was described in 1991. In particular, in recent years, robot-assisted laparoscopic partial nephrectomy (RALPN) has become the prevalent surgical option.

Partial nephrectomy, however, is underutilized for the surgical treatment of renal tumors among elderly patients. Reasons for this underutilization are not fully understood, but many urologic surgeons view increasing patient age as a relative contraindication to partial nephrectomy. Older and sicker patients undergo a radical nephrectomy to reduce the length of the operation and lower complication rates [32].

Partial nephrectomy, while saving part of the healthy kidney and preserving the organ function, presents possible severe complications. They include [33]:

- Urinary leakage. This is one of the most common complications of partial nephrectomy.
- Hemorrhage. It most commonly occurs during tumor resection, clamp removal, and the immediate or delayed postoperative period.
- Renal insufficiency. Renal vascular occlusion is often necessary during the partial nephrectomy to minimize hemorrhage during resection and to allow for adequate visualization for precise excision and repair. However, minimizing ischemia time is important for maximizing postoperative renal function.

Complications in partial nephrectomy depend mainly on damage to critical regions of the kidney such as the renal arteries and the many blood vessels that run through it. The damage to these structures can severely compromise the functionality of the kidney, causing the complications listed above, especially the hemorrhage. For these reasons, as in the case of prostatectomy, the success of the intervention entirely falls on the skill and experience of the surgeon.

Also for this surgical procedure, open surgery, laparoscopic surgery, and robot-assisted surgery are available.

2.5.1 Open surgery

With the open partial nephrectomy surgery, the kidney is accessed through a large incision (Fig. 2.10) which can be different according to the approach [34]:

- Retroperitoneal approach: the surgeon accesses the kidney through an incision on the flank of the patient.
- Transperitoneal approach: the surgeon accesses the kidney through a chevron incision, a cut made on the abdomen of the patient below the rib cage.

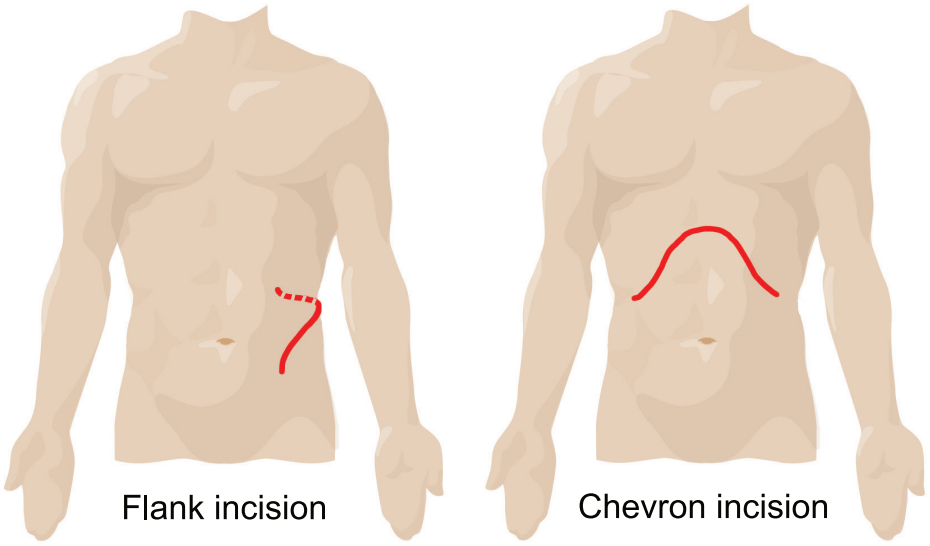


Figure 2.10: Open nephrectomy incisions [10]

2.5.2 Laparoscopic surgery

The most commonly used technique is the transperitoneal approach. As for radical prostatectomy, five small incisions are executed to place the trocars to access the kidney (Fig. 2.11):

- a 12-mm trocar is placed for the robotic camera situated 2 cm medial and inferior to the tip of the 12th rib.
- a pair of 8-mm robotic trocars are placed 2 cm inferior to the costal margin, and the other is placed 2 fingerbreadths superior to the iliac crest.
- a 5-mm port is inserted under the xiphisternum to allow retraction of the liver.
- a 12-mm umbilical trocar is used for the assistant's instruments.

Even in this case the pneumoperitoneum is created via a Veress needle placed at the umbilicus.

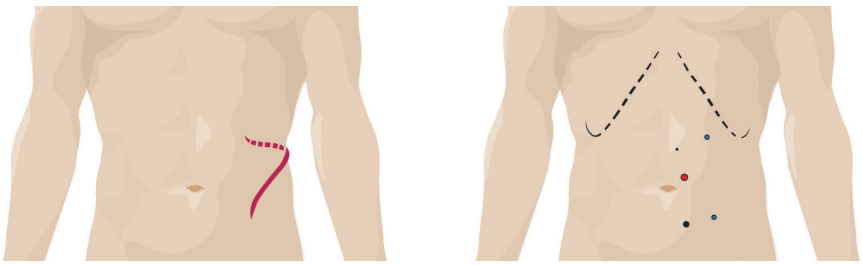


Figure 2.11: Open versus laparoscopic nephrectomy incisions [11]

2.5.3 Robot-assisted surgery

Robot assisted laparoscopic partial nephrectomy is similar to the classic laparoscopic nephrectomy, using small incisions on the patient's abdomen. The surgeon works on the patient through the surgical robot (e.g., da Vinci Surgical System), passing the robotic instruments and the endoscope through the tiny incisions on the abdomen.

2.5.4 Comparison of approaches

While a partial nephrectomy can be done through the open surgery or the laparoscopic surgery approaches, robotic kidney surgery offers apparently significant advantages over both of these approaches.

In particular, compared to the open surgery, the robotic kidney surgery exploits smaller incisions and doesn't require cutting through muscle or bone, resulting in less scarring and trauma to the patient and faster recovery time.

Operative outcomes

During the partial nephrectomy, renal vascular occlusion is often necessary to minimize hemorrhage during resection. Clamping of the renal artery, if long-lasting, can lead to renal ischemia (restriction in blood supply to tissues), compromising the functionality of the kidney. Minimizing warm ischemia time (WIT) is the primary objective for maximizing postoperative renal function. Ideal WIT is still under debate in the current literature, but

WIT < 30 min is recommended in order to reduce renal ischemic injury. Warm ischemia time (WIT) is shorter for RALPN than for LPN (20 minutes on average versus 28 minutes on average), and blood loss is significantly lower in RALPN. [35].

Instead, the OPN, while being the RALPN and LPN performed through smaller incisions, show a decrease in blood loss, while the ischemia times are approximately the same for OPN and RALPN [36].

Oncological outcomes

There are many reports on the oncological and functional outcomes after three years of observation following a RALPN, but there are few reports on the long-term outcomes (e.g., > 5 years) associated with RALPN. Furthermore, there are even fewer comparative studies of the oncological and functional outcomes of RALPN and LPN after five years of observation, while long-term oncological and functional outcomes from OPN and LPN are well established.

The only study, at the moment, which compare the long-term oncological outcomes of RALPN and LPN, reported that RALPN yields similar oncological and better functional long-term outcomes than LPN. Notably, the glomerular filtration rate (GFR) (the flow rate of filtered fluid through the kidney), the primary indicator of the kidney's condition, is significantly superior in patients with complicated cases who underwent RALPN than those who were treated with LPN [37].

Functional outcomes

When comparing OPN, RALPN, LPN, these techniques offer low morbidity and high success. The postoperative complications such as urinary leakage and hemorrhage are similar for the three techniques.

However, the median hospital stays for RALPN, LPN, and OPN are two days, two days, and three days, respectively [36].

2.6 Overview: research questions

The surgical removal of the localized prostate cancer and renal cancer continues to be the most definitive treatment of these diseases.

There is a trend toward consistently better outcomes following RALRP as opposed to LRP, and RALPN as opposed to LPN. RALRP and RALPN can successfully diminish the learning curve that surgeons face when beginning to perform LRP and LPN. They also offer quicker and superior return to continence and decreases in operative time as well as the length of the hospitalization. Moreover, with the robotic approach, the surgeon regains much of the degree of freedom for dissection that is lacking with LRP and LPN. Although economic considerations are vital, the advantages provided by robotic technology have the potential to minimize patient morbidity while improving both functional and oncological outcomes. As robotic technology evolves and becomes more prevalent, there is likely to be continued innovation and improved surgical outcomes. Ultimately, RALRP and RALPN are new technologies that deserve our attention and need further evaluation [17].

Despite this trend, current data suggest that results ultimately depend more on the surgical technique than on the surgical approach. Dissimilarity in outcomes among high-volume surgeons points toward distinctions in quality of care that are probably related to variations in surgical technique. Furthermore, regarding the radical prostatectomy surgery, rates of blood loss, incontinence, and erectile dysfunction vary widely from surgeon to surgeon. The same reasoning can be applied to the partial nephrectomy surgery. It is clear that the best chance for cure rests in the most experienced hands.

In this regard, an intraoperative robotic system, which bridges the gap between less experienced surgeons and their senior counterparts in terms of experience and dexterity, becomes of fundamental importance. This system must prevent an excessive risk of damage to fragile structures such as veins, arteries, and nerves, with the following bleeding problems during the operation and postoperative dysfunction for the patient, and should be exploited for any minimally invasive procedure performed with the da Vinci Surgical System.

2.7 Current state of art

Computer-assisted technologies, coupled with robotic surgical systems, can enhance surgeons' capabilities by providing additional information regarding the surgical gestures.

The intraoperative identification of vessels to be preserved has been explored by Shinji Onda *et al.* [3] using preoperative information registered on the patient and visualized employing augmented reality.

Their study reports the utility of early identification of the inferior pancreaticoduodenal artery (IPDA) using an augmented reality-based navigation system.

They fused the images obtained by preoperative computed tomography with a real-time operative field image, displayed on 3D monitors. The reconstructed vascular images and the inferior pancreaticoduodenal artery were visualized to facilitate image-guided surgical procedures [3] (Fig. 2.12).

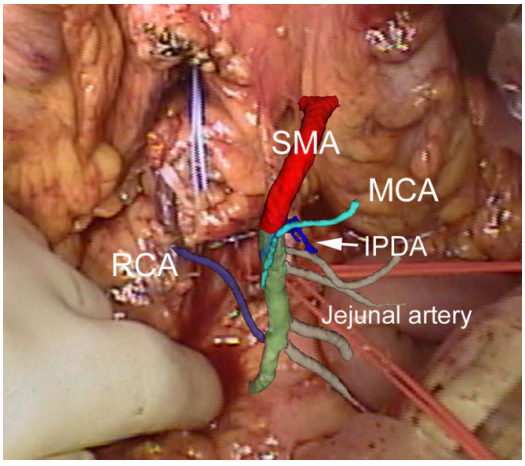


Figure 2.12: Augmented reality-based navigation system [12]

However, this approach has to deal with the dynamic changes in the anatomy between the data acquisition phase and the actual surgical procedure. In fact, these changes can frequently occur due to [5]:

– different pose of the patient with respect to the one in which the preoperative information was stored.

– CO2 abdominal insufflation that presses and changes the shape of the organs.

– instrument tissue interaction, heartbeat, and breathing that affect the registration on a smaller scale.

In order to measure the intraoperative tissue movements, computer vision and image processing algorithms have been exploited to track soft tissue area relying only on the image characteristics. Early works on soft tissue tracking algorithms applied to endoscopic images have been done exploiting optical flow techniques [4]. Stoyanov used scene flow estimation techniques for the recovery of 3D structure and motion of the operating field from stereoscopic images, propagating this information to obtain a denser surface deformation identification. The main advantages of such methods are the sub-pixel accuracy and low execution time. However, for long-term endoscopic videos, the tissue area appearance may change or can be partially or wholly occluded by instruments or camera movements. For these reasons, such algorithms typically accumulate errors resulting in tracking drift or fail in case of occlusion [38].

Although the methods introduced have reached high levels of performance, they lack specific characteristics, such as high accuracy, adaptation to changes and adverse circumstances, and real-time processing, which are fundamental to the intended purpose.

Furthermore, the methods presented have focused only on the intraoperative identification of the anatomical structures to be preserved. In fact, the implementation of intraoperative safety constraints, which allow steering the robotic tool away from the fragile and critical structures, has not been yet appropriately realized.

This thesis work, therefore, introduces a system which seeks to overcome the limits of the methods above, providing a useful and innovative solution for the field of minimally invasive robotic surgery.

Chapter 3

Robot-assisted surgery: a novel intraoperative system

3.1 Goals

The present thesis work was carried out at the Neuroengineering and Medical Robotics Laboratory (NearLab), department of "Elettronica, Informazione e Bioingegneria" (DEIB). The project aims at developing a new intraoperative solution for minimally invasive robotic surgery. This project was developed for the da Vinci System, but other robotic systems can easily exploit it.

The presented method tries to overcome the current limits of robotic surgery which, although highly innovative, maintains a high risk of damage to fragile and dangerous regions such as vessels, arteries, and nerves. This leaves the management of the main risk factors entirely in the surgeon's dexterity and experience due to the lack of the force feedback.

Moreover, this method attempts to overcome the limits found in previous works which, to preserve the above mentioned sensitive regions, exploit the image-based surgery approach to track the surgical instruments in conjunction with preoperative images, in order to guide the procedure. These methods are penalized by not being able to consider the dynamic changes that can occur on anatomical structures in the time interval between the preoperative acquisition phase and the actual surgical procedure. Moreover, through the

augmented reality, they provide only a visual guide to the surgeon, without defining any force feedback that can prevent him from damaging the fragile and critical anatomical structures.

3.2 Methods and innovations

Considering the observations of the previous paragraph, an intraoperative active constraints (AC) definition system was developed in this thesis work. The logical functioning of the solution devised is shown in Figure 3.1, which introduces the essential elements of the system.

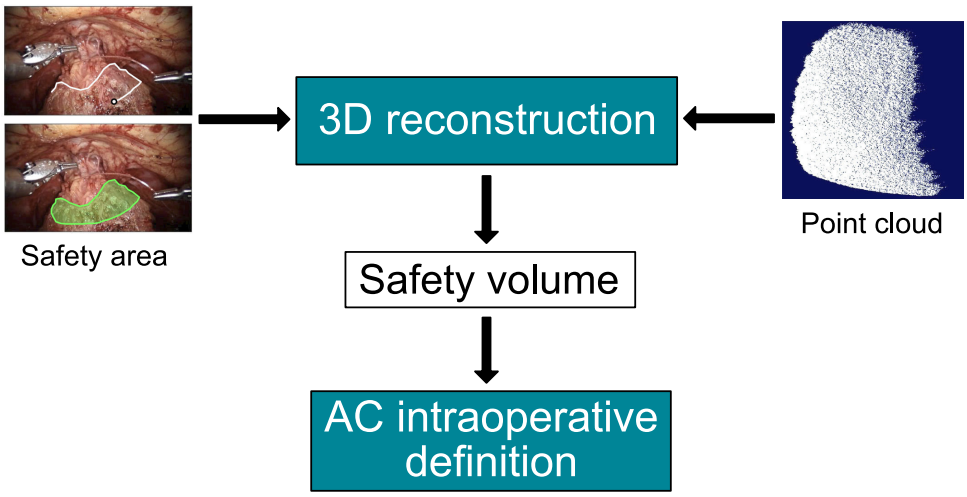


Figure 3.1: Operating scheme of the system

The system designed allows the surgeon to define an intraoperative safety area on the patient's anatomical structures. This is achieved by moving a pointer, shown on the surgical console screen, with the same master controllers the surgeon uses to perform the surgical procedure (Fig. 3.2).

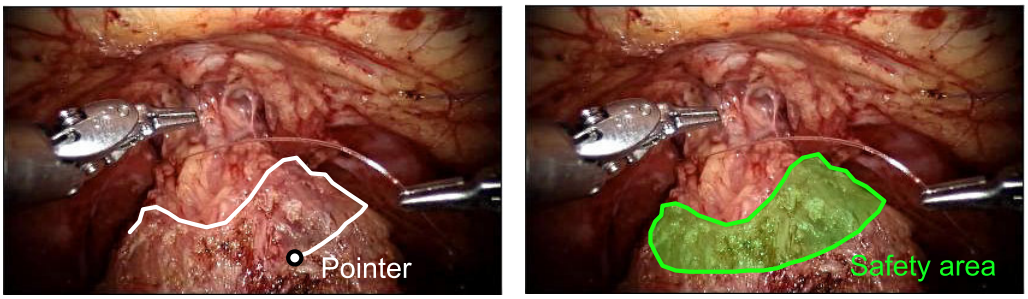


Figure 3.2: By moving the cursor, the surgeon can define the safety area freehand [13].

During the execution of the system, a point cloud of the patient's anatomical structures is produced by the disparity map between the images of a stereoscopic camera (e.g., the da Vinci endoscope).

The safety area is employed to identify the portion of the point cloud belonging to the anatomical structure that the surgeon intends to select. By exploiting the 3D information of the point cloud, the safety area is then modeled to recreate a safety volume and its surface (Fig. 3.3).

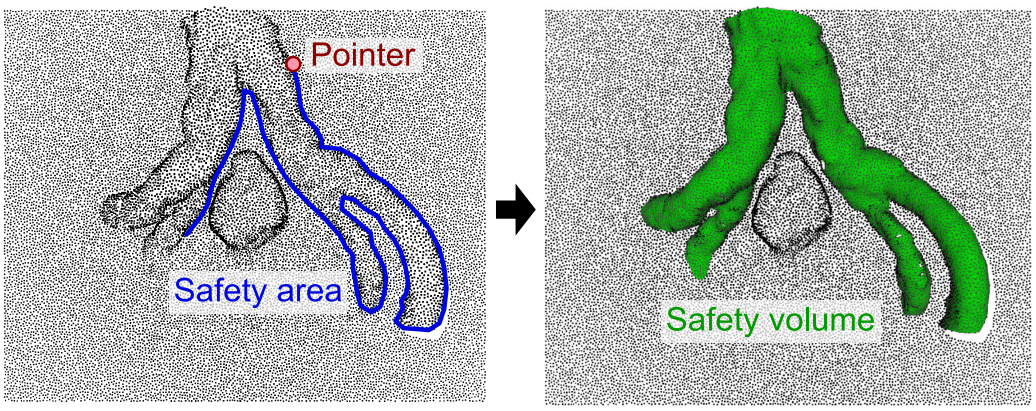


Figure 3.3: The selected point cloud is exploited to reconstruct the safety volume.

The safety volume, which represents a region of space within which the robotic tool should not enter, is finally exploited to define the active constraints (also known as safety constraints). In this way, a repulsive force is

generated, which prevents the robotic tool from entering the constrained region, thus avoiding damage to the fragile and sensitive anatomical structures selected (Fig. 3.4).

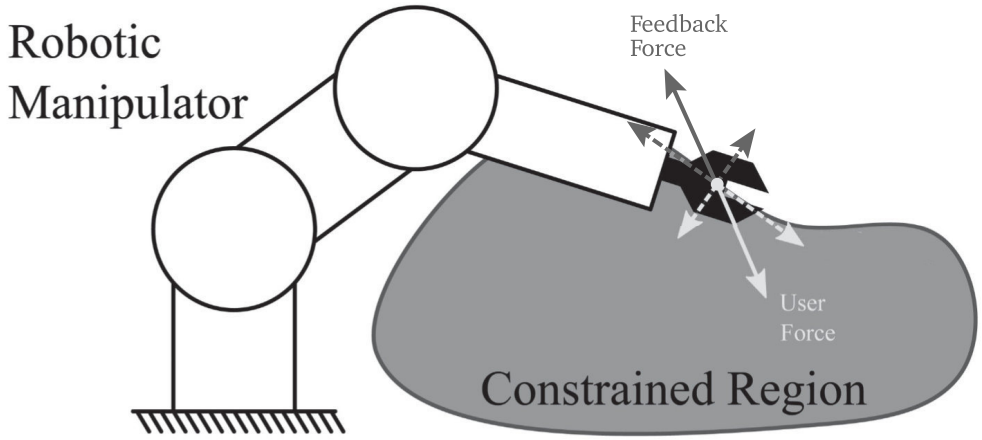


Figure 3.4: The constrained region generates a repulsive force that prevents the robotic tool from entering it [14].

The system conceived thus allows overcoming the restrictions that characterize the previously introduced methods. First, the ability to define a safety area intraoperatively eliminates the problems caused by the registration of preoperative images on the surgical scene. In fact, if dynamic changes of the anatomical structures modify the selected region, the surgeon can freely define a new safety area, which is processed instantaneously to generate a new safety volume. The same reasoning can be applied in case of camera movements or occlusions. Moreover, thanks to the definition of the safety constraints, the surgeon will not only have to rely on the visual feedback given by the augmented reality but will also be able to enjoy the force feedback that will guide him during the surgical procedure.

To summarize, the system can be divided into two main parts:

- the surgeon draws the safety area freehand moving the pointer through the master joysticks at the da Vinci console.

- the active constraints system generates repulsive forces on the master joysticks depending on the robotic tool position relative to the safety volume.

In the next sections, the active constraints and their implementation will be introduced. Afterward, the architecture of the system will be shown, as well as the various modules that compose it, explaining the methods and strategies used to implement them and the motivations that led to the various design choices.

3.3 Active constraints

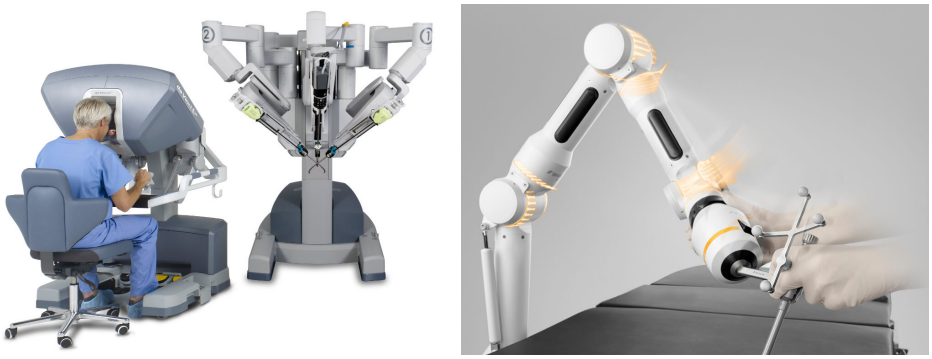
Robotic assistance allows surgeons to perform quick and tremor-free procedures. Human error, however, is not eliminated, since the surgeon can inadvertently hit sensitive anatomical areas, maintaining a high risk for the patient. Under these conditions, the risk of unexpected collisions between the robotic instrument and the surrounding areas remains high, especially in restricted areas and close to fragile elements such as arteries, veins, and nerves. An increase in safety and a reduction in the mental load of the surgeon have been achieved through the generation of active constraints (virtual fixtures) which can, for example, act as a safety layer to prevent the surgical instrument from entering a restricted area, even if this contradicts the commands given by the surgeon. The active constraints can, therefore, be used as the primary tool to avoid collisions and the resulting damages. This topic represents a fundamental part of this thesis work, within which a simple but effective proximity constraint is proposed that ensures that the instruments do not enter restricted regions.

Active constraints can be defined as "control strategies", which can be used in human manipulation tasks to improve or assist the surgical procedure by anisotropically regulating motion [39]. This is achieved by comparing the robotic tool position with respect to known restricted regions or pre-planned trajectories, and then attenuating or nullifying the user command, which will cause the manipulator to deviate from a plan, or enter a restricted region.

An in-depth survey of active constraints was presented by Bowyer *et al* [39], which listed the main properties of active constraints, as well as their implementations. In the next sections, these properties will be introduced, arguing the reasons that led to the current implementation of active constraints in this thesis work.

Active constraints properties

Depending on how the user interacts with the device that implements active constraints, two main categories of devices can be identified:



(a) (b)

Figure 3.5: (a) Teleoperated and (b) hands-on robot [15]

– "Teleoperated" devices implement a "master-slave" configuration. The user is physically separated from the robotic manipulator (slave) but controls it via the command console (master) (Fig. 3.5a). Telemanipulation offers benefits such as operation in restricted and unsafe environments and motion scaling. However, any environmental information that the user needs, to perform the procedure, has to be interpreted at the slave device, transmitted to the master, and then presented in some way.

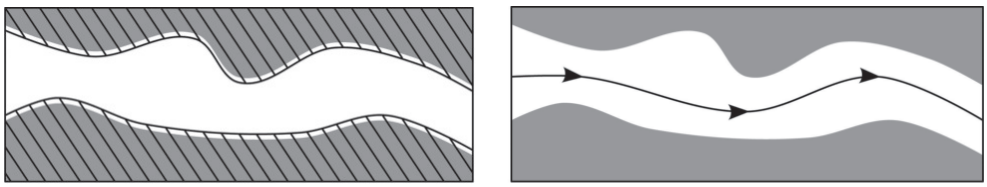
– "Hands-on" devices are those where the human user interacts directly with the tool-carrying manipulator (Fig. 3.5b). In this case, the user can feel the tool's interaction with the environment, integrating the

user into the procedure. However, since the user is holding the robotic tool, improved access and motion scaling cannot be provided.

As already shown at the beginning of this thesis, the da Vinci Surgical System falls within the first category.

"Regional" constraints (also called "forbidden region virtual fixtures") bound the manipulator's tool to specific regions within its task or joint space (Fig. 3.6a). In other words, this strategy prevents the manipulator tool from entering certain areas. This type of active constraints offers benefits such as task simplification [40] [41] and, especially, reduced risk of tools damaging protected regions [42].

"Guidance" constraints (also called "guidance virtual fixtures") make the user follow a pre-established path or move toward a specific target [43] [44] (Fig. 3.6b).



(a) (b)

Figure 3.6: (a) Regional and (b) guidance constraints. In regional constraints the user is prohibited from penetrating the hashed area [14].

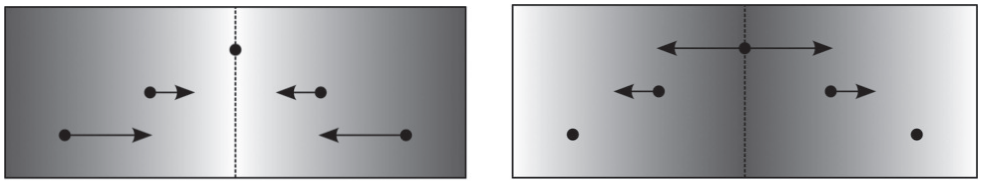
Due to the more intrusive nature of the guidance constraints with respect to the regional constraints, it was decided to implement the latter in the following work. It is, however, important to point out that the distinction between regional and guidance constraints is very subtle in case the regional constraints are very restricted.

The active constraints act can also be considered as either attractive (encourages motion toward the constraint) or repulsive (encourages motion away from the constraint) [45].

The key difference between these two categories is where active constraints come into action. With repulsive constraints, the user's motion is only modu-

lated once the tool has already passed into the restricted region. Conversely, for attractive constraints, the modulation comes into effect before the robotic tool reaches the selected region.

Examples of these two constraint types are shown in Figure 3.7.

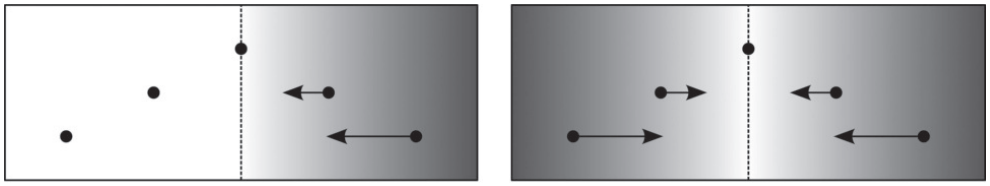


(a) (b)

Figure 3.7: (a) Attractive and (b) repulsive constraints [14]

Since the purpose of the intraoperative system created in this work is to prevent the surgeon from entering a security zone and not being guided towards a target or a pre-planned path, the actual constraints implemented are repulsive.

The last property of active constraints listed is referred to their directionality, which can be unilateral (acting on only one side) or bilateral (acting on both sides) [46].



(a) (b)

Figure 3.8: (a) Unilateral and (b) bilateral constraints [14]

Conventionally, a regional constraint would be constructed from unilateral surfaces, which prevent tool motion into the restricted space, whereas a guidance constraint would typically be bilateral, always encouraging tool motion toward the guidance path or point. Examples of these two constraint types are shown in Figure 3.8.

The constraints implemented, as shown at the beginning of this section, are regional, so they are also unilateral.

Generalized active constraints framework

To present the active constraints field systematically, a generalized framework is first introduced. This framework is shown in Figure 3.9, and is based on three primary processes, which are required in all practical active constraint implementations.

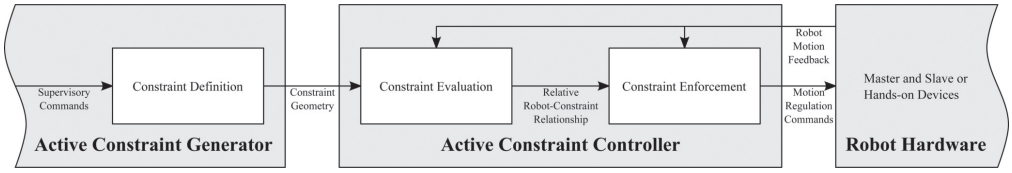


Figure 3.9: Generalized active constraint implementation framework [14]

The first stage in applying an active constraint is the input of the geometry, which will constitute the constraint itself. In the generalized active constraint framework, this task is carried out by the "constraint definition" module. The input to this module can be via human user supervision or autonomous sensing, and the output will be some computational representation of the geometry. In the case of this current work, the output is represented by the selected point cloud, as explained in the previous sections.

Before any constraint can be enforced, the relative configuration of the constraint and the robot must first be evaluated so that the anisotropy in the motion regulation can be established. This relative configuration can take a variety of forms; however, it is typically a pair of colliding or closest points, with one on the constraint and one on the robot. This stage of the active constraint computation is carried out by the "constraint evaluation" module.

Once the relative configuration of the constraint geometry and the robot is known, the active constraint can be applied. This is undertaken by the "constraint enforcement" module, which converts the relative robot-constraint

configuration into hardware-specific commands, which will regulate the motion of the human user [39].

Constraints definition methods

Three-dimensional (3-D) reconstruction of surfaces is a general method for generating active constraints around physical objects within a robot's workspace. However, the use of unreconstructed raw data (i.e., point clouds) is currently more widespread within the literature.

The chosen constraint representation directly affects the geometrical form that constraints can take and the subsequent choice of the constraint evaluation method.

There are various methods for computationally representing active constraints. Simple methods use points, lines, and planes, while more complex methods exploit polygonal meshes and parametric surfaces.

Polygonal meshes provide great flexibility for representing constraint geometries (Fig. 3.10), but they are considerably more complicated to construct, evaluate, and store than the more simple representations.

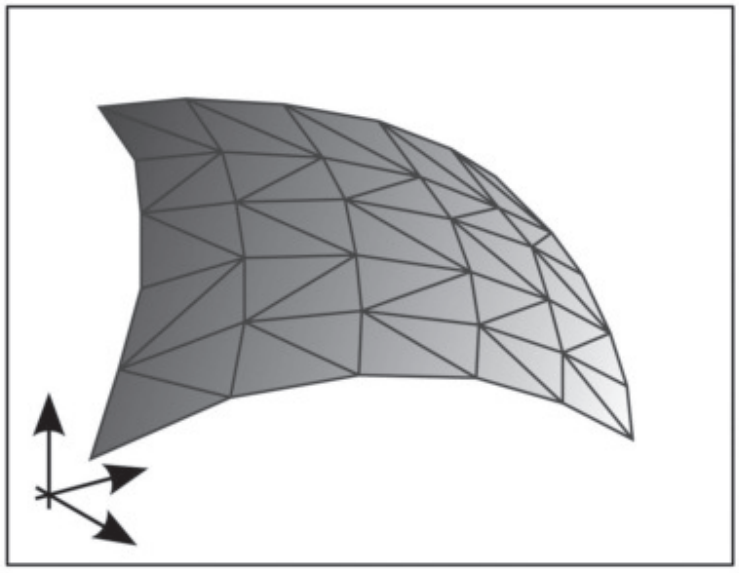


Figure 3.10: Polygonal mesh constraints representation [14]

For some applications, the degree of surface conformity offered by meshes is not worth the increased complexity. For this reason, it was decided to implement the active constraints based on the selected points of the point cloud.

So, by sampling a set of cartesian points on the surface, the representative point cloud can be constructed (Fig. 3.11). As previously shown, this can be produced merely from 3-D scanners, range cameras, or stereoscopic cameras, and can be maintained effectively in real time, thus representing a useful tool for defining the active constraints.

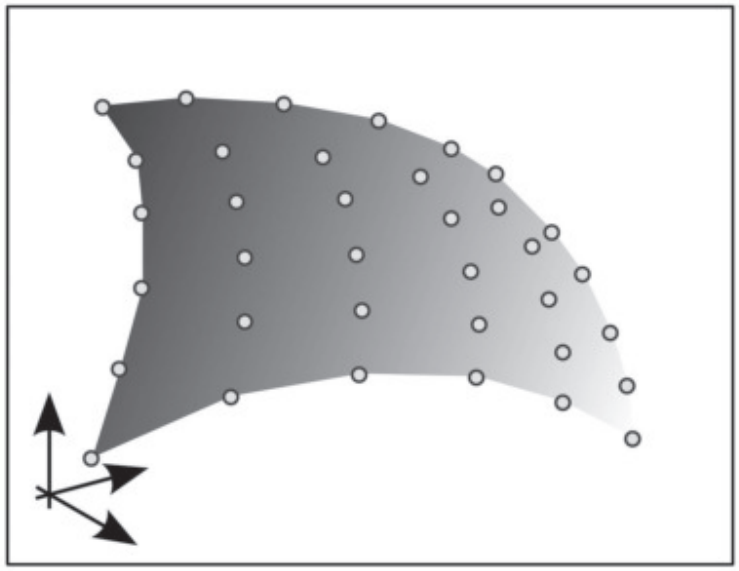


Figure 3.11: Point cloud constraints representation [14]

Constraints evaluation method

For the implementation of active constraints, a precise definition of the relationship between the robotic tool and the constrained region is necessary. Conventionally, the geometric relationship of interest is the proximity between the robot's tooltip and the active constraint geometry, or just whether the two have collided.

Constraints enforcement method

There is a wide variety of methods that are described in the literature for enforcing active constraints. In general, the input to these methods is the current pose of the robotic tool and the desired pose when it collides with the constrained region. Based on this relative configuration, the active constraint enforcement module decides if, and how, anisotropic motion regulation should be applied. The several approaches for enforcing active constraints defined in literature have significant differences between them and their various attributes make them more suitable for some applications than others.

The most common constraint enforcement methods described in the literature are those where the effect of a constraint on a device is computed using a simple function of the proximity between the robotic tool and the constrained region. Linear functions, generally based on minimum constraint proximity, have been widely used to enforce both regional and guidance constraints. The mechanical analogy of an elastic spring between the robotic tool and constraint (with stiffness k_p) is often used when describing linear functions, as is the classic control concept of a proportional position controller. A linear function for a Cartesian pure spring impedance controller takes the form

$$\mathbf{f}_p = k_p(\mathbf{p}_d - \mathbf{p}_c) \tag{3.1}$$

where $\mathbf{f}_p$ is the constraint force vector, k_p is a linear gain, $\mathbf{p}_d$ is the desired tool position, and $\mathbf{p}_c$ is the current tool position. The desired tool position is represented by the point on the constrained region surface closest to the robotic tool when the latter collides or is barely inside the constrained region. Alternating the value of k_p between positive and negative changes the constraint between attractive and repulsive, respectively, and changing its magnitude affects how strongly the constraint acts upon the robot and a piecewise function can be incorporated to implement unilateral constraints.

In this work, derivative terms have been used to extend linear enforcement functions and improve their properties. If a user moves against a constraint

at a higher velocity differential, then it will take a more considerable force from the controller to slow them down and minimize any non-conformance. By introducing a derivative (or viscous) term to the enforcement function, the controller can respond to this situation. In this case the previous equation takes the form

$$\mathbf{f}_p = k_p(\mathbf{p}_d - \mathbf{p}_c) + k_d(\dot{\mathbf{p}}_d - \dot{\mathbf{p}}_c) \tag{3.2}$$

where $\mathbf{f}_p$ is again the constraint force vector, and k_d is a derivative gain. If the constraint geometry in this formulation is static (i.e., $\dot{\mathbf{p}}_d = 0$), then the derivative term will be isotropic so that it opposes motion conforming to the constraint as much as it opposes motion that does not. If $\dot{\mathbf{p}}_d$ is set to move, along a pathway, for example, then the robot will be constrained to follow its motion.

The active constraints, therefore, generate a repulsive force on the robotic tool when it tries to enter the safety zone, thanks to a virtual linkage (Fig. 3.12), which is typically elastic or viscoelastic (as in this case). If the robotic tool violates the constraint, then the desired tool pose will "remain" on the surface, causing them to separate and creating the virtual linkage, which finally generates the repulsive force.

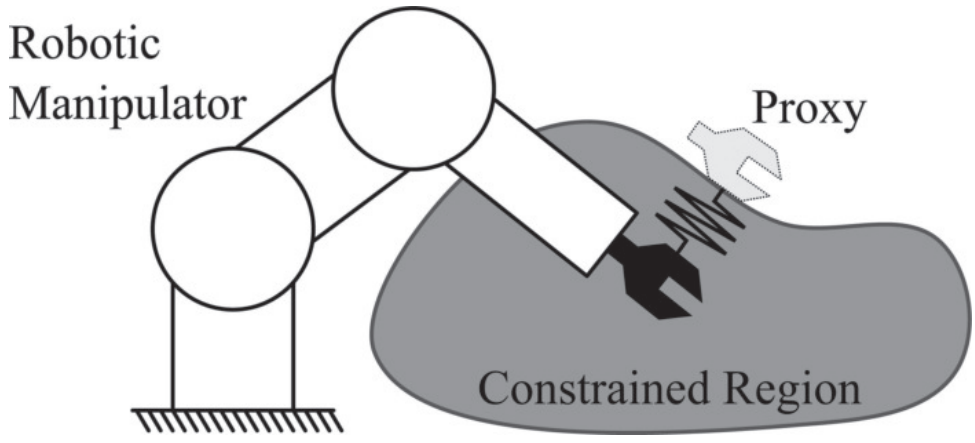


Figure 3.12: Proximity constraints enforcement [14]

3.4 The system architecture

For writing the software, it was decided to use the C++ programming language. C++ is a general-purpose programming language which has imperative, object-oriented and generic programming features, while also providing facilities for low-level memory manipulation. Given its efficiency, flexibility and, above all, its compatibility with the necessary libraries, which will be exposed later, it was considered the most appropriate and logical choice for the drafting of the code.

The software was developed within an innovative package called Assisted Teleoperation with Augmented Reality (ATAR) which can be used to design interactive augmented reality or virtual reality tasks using a stereo camera and master-slave manipulators [47].

The system was tested on a da Vinci Research Kit (dVRK) that comprises five manipulators (two masters, two slaves, and a camera arm), a stereo endoscope and stereo vision console.

The main libraries used to develop the system are:

- ROS (Robot Operating System): libraries and tools which help to create robot applications [48].
- Qt GUI: a module which provides classes for windowing system integration, event handling, OpenGL and OpenGL ES integration, 2D graphics, basic imaging, fonts, and text [49].
- Bullet physics: a physics library which simulates collision detection, soft and rigid body dynamics [50].
- Visualization Toolkit (VTK): a software system for 3D computer graphics, image processing, and visualization [51].
- The Point Cloud Library (PCL): a standalone, large scale, open project for 2D/3D image and point cloud processing [52].

The camera images and the manipulator poses are read through ROS topics, the channels where nodes are subscribed for to read and publish messages, while the QT GUI module is exploited to create the graphical user

interface. Bullet Physics library is used for the rigid and soft body dynamics simulation and graphics are generated using the Visualization Toolkit (VTK). Finally, the point cloud is managed and processed with the Point Cloud Library (PCL).

The package consists of several classes and methods that, by following a predetermined flow of actions, allows the user to build tasks and scenarios with the preferred configurations (Fig. 3.13).

Creating a task

TaskHandler is the class that subscribes to a control events topics (published by the GUI) and loads and unloads tasks. When a new *Task* class is defined, it is possible to configure it with different rendering configurations (e.g., augmented reality or virtual reality) and different manipulators. The augmented reality configuration lets to superimpose virtual elements and objects to the actual images taken by the camera, while the virtual reality configuration is used to create a virtual 3D environment. The block diagram in Figure 3.13 shows the objects of a task class, where the main elements are the *Rendering*, *SimObjects* and *Manipulator* classes, which are explained in the following paragraphs.

The usual flow consists in creating *SimObjects* and adding them to the world in the constructor. Two periodically called functions can be used to update the state of these objects and write task logic and other things that need to be done at runtime:

- *TaskLoop*: this is called from the main thread at a refresh rate of about 30 Hz which makes it the choice for updating graphics and task-related logic.

- *HapticsThread*: this method is called from a separate thread, and its refresh rate can be set directly in the method. This thread exists for haptics related matters where a high refresh rate is needed to provide stable feedback.

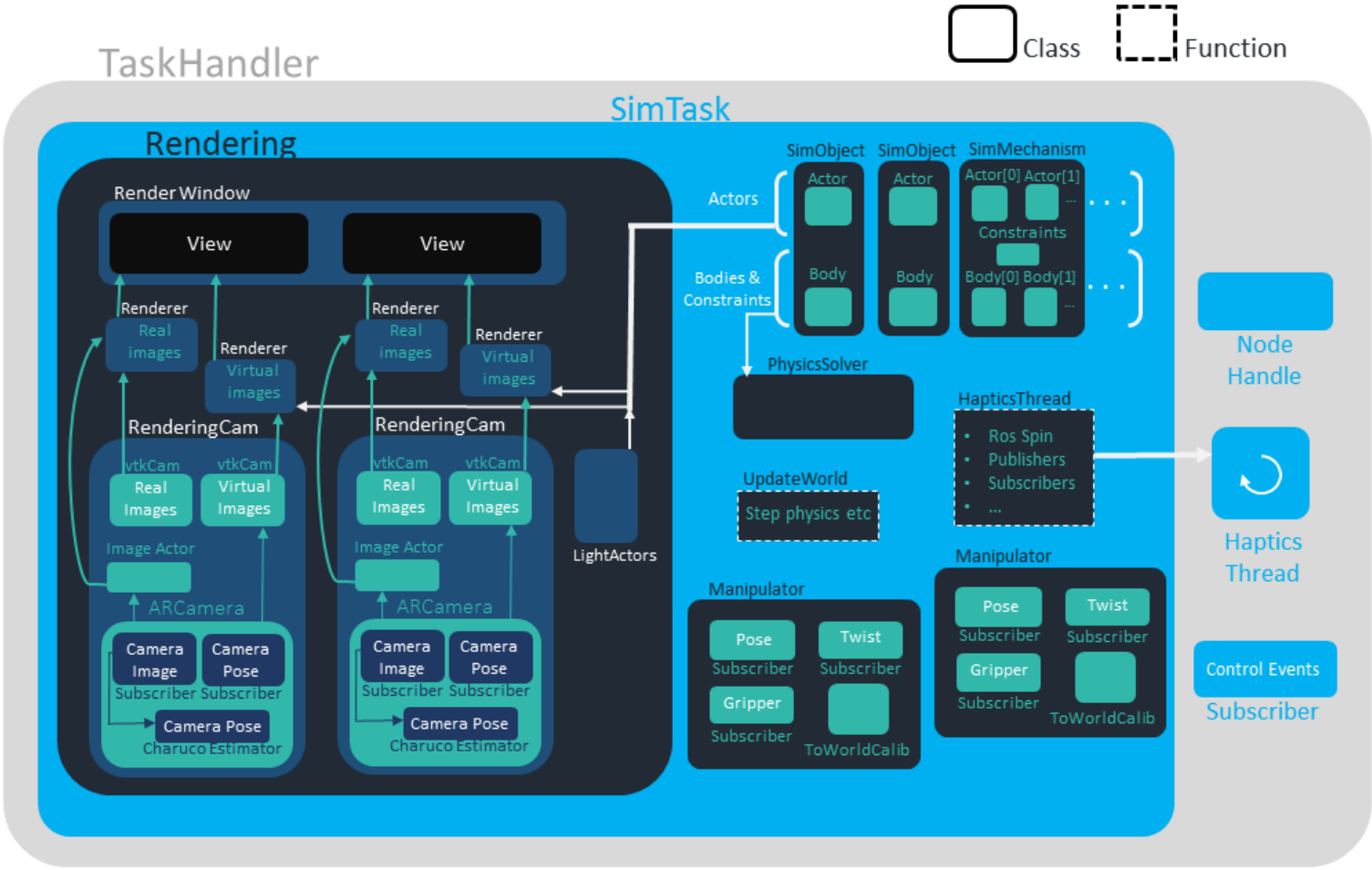


Figure 3.13: ATAR block diagram [16]

Rendering class

As its name suggests, this is the class which produces the graphics. The constructor of the *Rendering* class lets the user specify the configuration of the windows desired. In the dVRK setup, the surgeon has two displays in the user console (one for each eye), and a third optional display is set for other viewers in the room. These three displays are connected to the same graphics card in a horizontal layout as the Figure 3.14 shows.

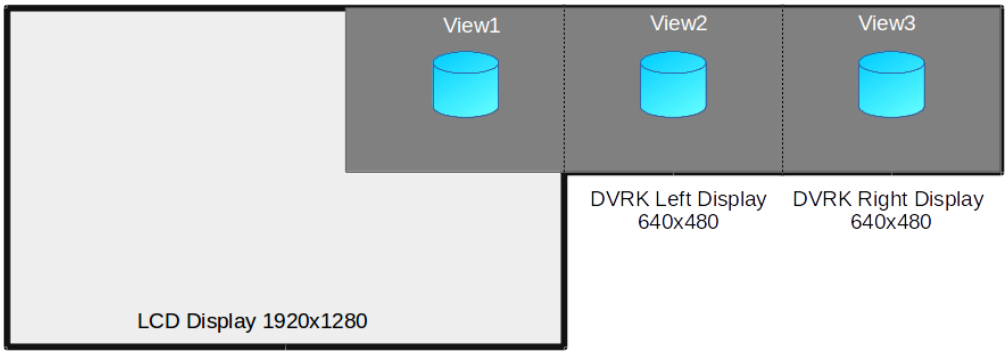


Figure 3.14: Rendering windows [16]

The user can set the rendering windows as shown in the appendix A.1.

SimObjects class

Objects are defined using the *SimObject* class. A *SimObject* can be dynamic (i.e., its pose will be updated by physics simulation), static (i.e., its pose is constant), kinematic (i.e., its pose is assigned externally, e.g., from a master device). There are some primitive shapes available (cube, sphere, cylinder, cone, and plane), but more usefully the shape can be from a .obj mesh file. After constructing a *SimObject*, the *AddSimObjectToTask* method let the user add the object to the simulation.

Meshes are decomposed into approximated compound meshes which can take up to a few minutes. So, to speed up the times, this process is done only once, saving the compound mesh in a separate file. The next time the

application is executed, it looks for the generated file and, if found, it is used, and the compound mesh generation is not repeated.

Manipulator class

The system, to interact with the virtual environment, needs to have an input device of some sort. The *Manipulator* class reads the Cartesian pose and twist of that device (assuming some other node is publishing them) and transform them into the virtual world reference frame. The *SimMechanism* class creates a virtual tool that follows the pose of the real manipulator.

AR Camera class

This class, used to interface with a camera through ROS, is similar to what the *Manipulator* class is for a robot arm. It reads images and the intrinsic camera parameters and, in case nothing is found on topics for the latter, it calculates them.

1. **Reading images**
   This uses *image_transport*, which provides classes and nodes for transporting images, to read images that are published on a topic, assuming that an external node is publishing them.

2. **Intrinsic calibration**
   Intrinsic camera parameters are needed for mapping the 3D world to the camera images. This perspective projection is modeled by the ideal pinhole camera, illustrated in Figure 3.15.

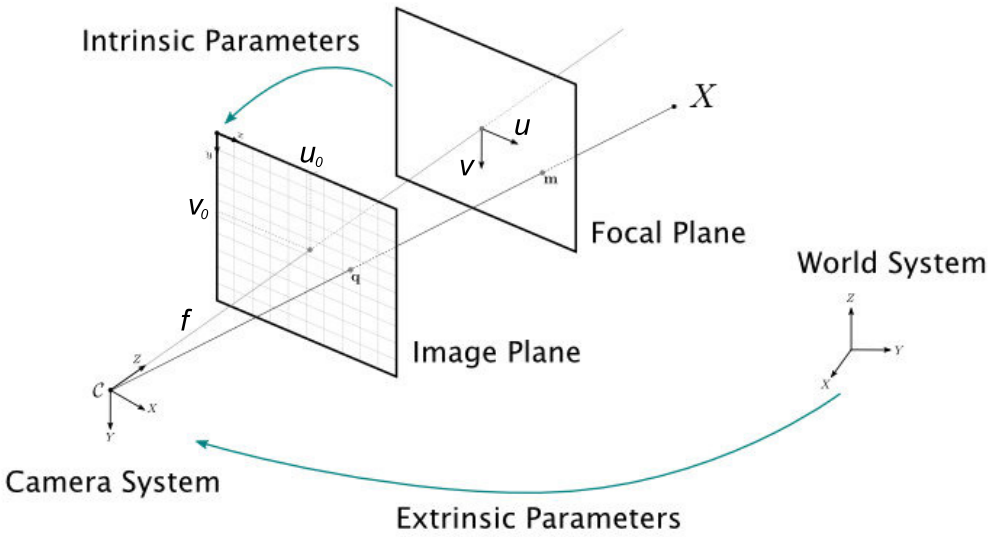


Figure 3.15: The pinhole camera model [17]

Hartley and Zisserman parameterize the intrinsic camera matrix [53] as

$$K = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

where each intrinsic parameter describes a geometric property of the camera:

- The focal length (f_x, f_y) is the distance between the pinhole and the image plane.
- The camera's principal axis is the line perpendicular to the image plane that passes through the pinhole. Its intersection with the image plane is referred to as the principal point. The principal point offset (u_0, v_0) is the location of the principal point relative to the film's origin.
- Axis skew (s) causes shear distortion in the projected image.

Within the system, these parameters can be obtained in three different ways:

- If the camera is already calibrated, through a yaml file (YAML Ain't Markup Language). This kind of file is a human-readable data serialization language and it is commonly used for configuration files.
- Subscribe to the *camera_info* topic associated with the camera. It is the main ROS way of accessing intrinsic parameters.
- If *ARCamera* does not find the parameters from any of the above two methods, it starts the intrinsic calibration procedure with a Charuco board, a combination of a standard chessboard with a set of Aruco markers. "ArUco" markers and boards are handy due to their fast detection and their versatility. However, one of the problems of "ArUco" markers is that the accuracy of their corner estimation is relatively low, even after applying subpixel refinement. On the contrary, the corners of chessboard patterns can be refined more accurately since two black squares surround each corner. However, finding a chessboard pattern is not as versatile as finding an "ArUco" board: it has to be completely visible, and occlusions are not permitted. A "ChArUco" board tries to combine the benefits of these two approaches (Fig. 3.16).

  Using the OpenCV Aruco library, the board markers can be detected in the camera image (Fig. 3.17) and intrinsic calibration, based on the markers'spatial information, can be computed.

  The user needs to take at least 15 different poses of the board in front of the camera (the instructions are shown in the image). If the calibration is successful, a yaml file is created with the corresponding calibration.

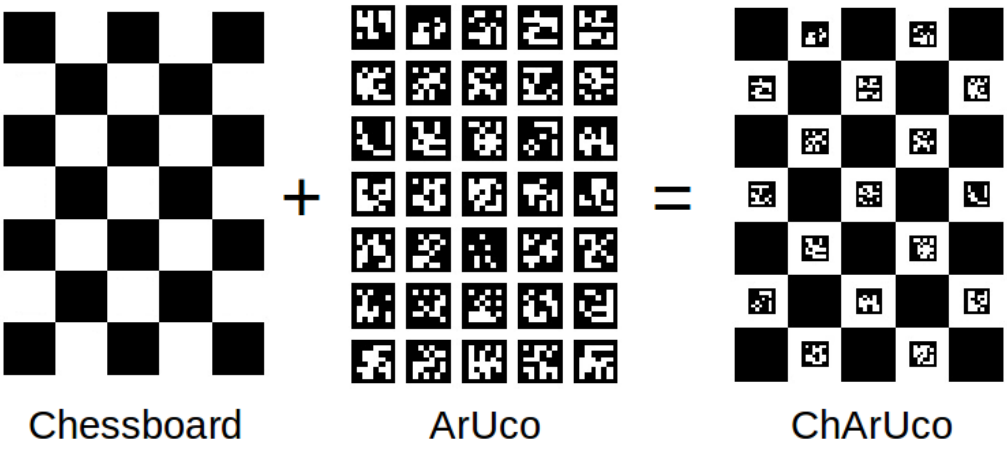


Figure 3.16: Charuco board [17]

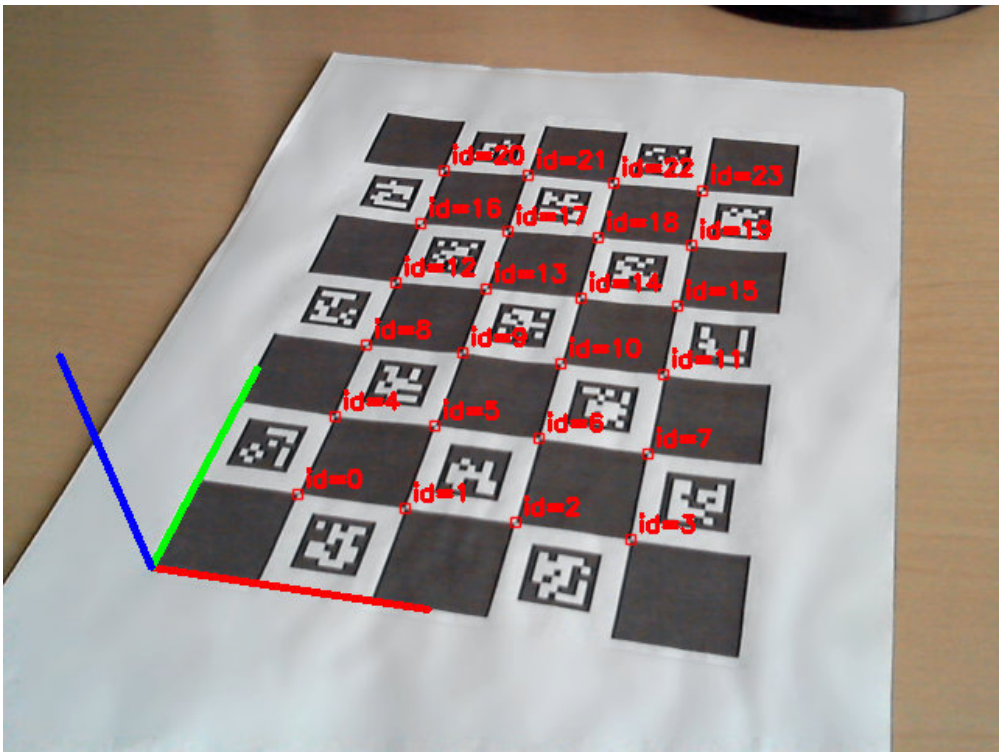


Figure 3.17: Charuco board detection [17]

After listing the primary methods and classes of the package, on which the system is based, the next sections will show the fundamental parts of the algorithm that constitute the operating flow of the system.

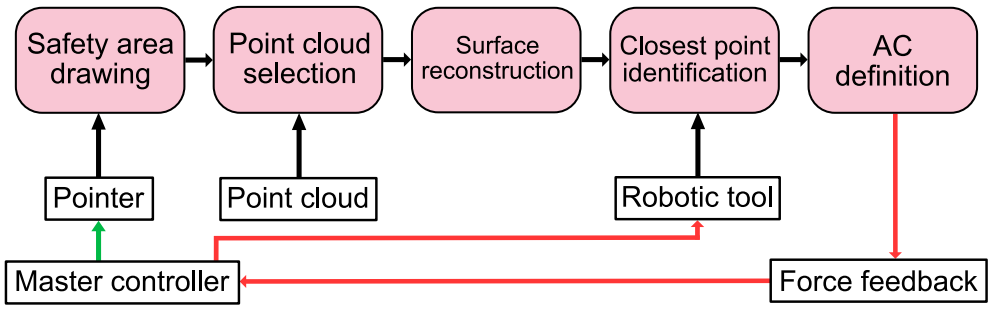


Figure 3.18: The system operating flow. When the master controllers are used to move the pointer (green path), the force feedback is deactivated as is the control of the robotic tool (red path).

To summarize, as shown in Figure 3.18, after the definition of the safety area through the pointer, the corresponding point cloud is selected. It is exploited to reconstruct the surface and to identify the point on it closest to the robotic tool. This point is finally used to calculate the repulsive force acting on the master controller as defined by the active constraints.

3.4.1 The pointer

The pointer allows the surgeon, who is manipulating the master arms at the da Vinci console, to draw the safety area on the image he sees through the stereoscopic screen.

The safety area is of fundamental importance for the reconstruction of 3D information, which in turn allows the calculation of the force that will oppose a movement towards the inside of the safety volume, thus ensuring the surgeon secure force feedback. In this regard, the pointer must guarantee the user easy maneuverability through the da Vinci master arms, to be able to draw a precise safety area.

At first, the workspace of the right master arm was measured at the surgical console to map the arm pose to the screen coordinates of the rendering windows (Fig. 3.19).

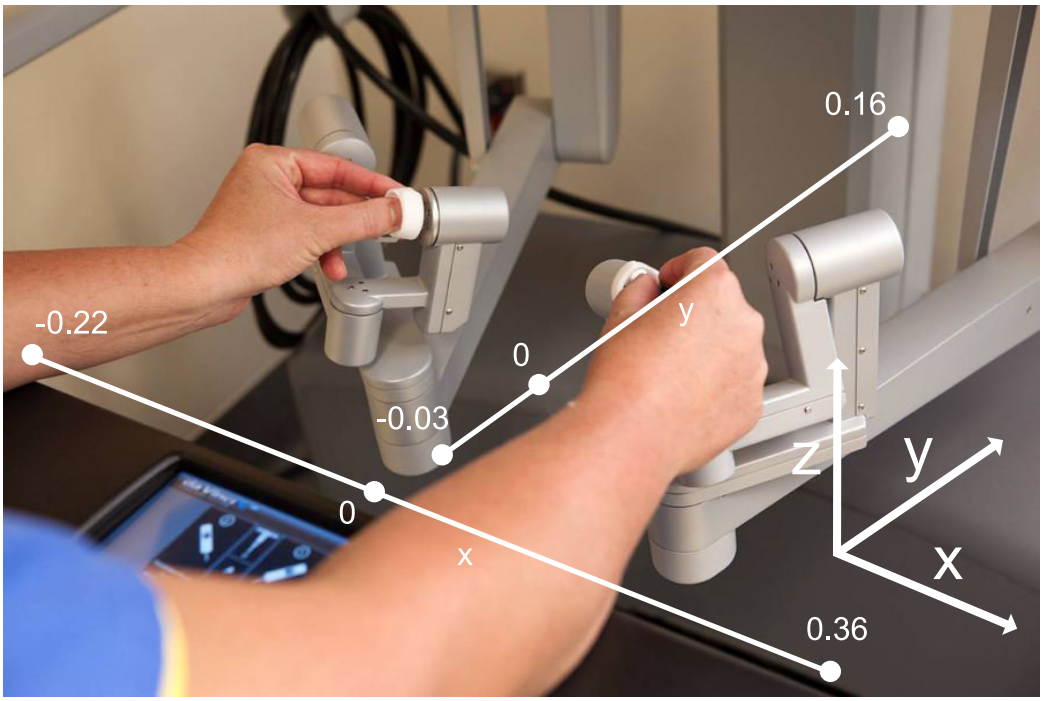


Figure 3.19: Master arms axes [18]

Only the x and y-axes were considered, choosing them as the movement axes for the pointer, while the z-axis was ignored, being the pointer represented in 2D coordinates. This procedure was repeated for the left arm, letting the surgeon chose which arm to use at the launch of the system, in case he is right-handed or left-handed. These workspaces were not measured up to the limits of movement of the arms, but in such a way to ensure comfortable and fluid maneuverability, thus preventing the surgeon from having to make large movements to reach the limits of the image with the pointer.

Given the above considerations, the workspaces obtained are in meters:

$$-0.22 < x < 0.1 \quad \& \quad -0.03 < y < 0.16$$

for the left master arm and

$$0.04 < x < 0.36 \quad \& \quad -0.03 < y < 0.16$$

for the right master arm.

At this point, these workspaces were used to transform the coordinate system of the da Vinci master arms into a symmetrical coordinate system, arbitrarily defined between -1 and 1 for both the x-axis and the y-axis. This passage has been realized both to simplify the implementation of the following steps and to provide, to those who need it, a more comfortable joystick coordinate system. This is achieved through an appropriate linear conversion as shown in the appendix A.2.

In this way, each master arm position is converted into a new position in the newly defined coordinate system. The choice of obtaining a coordinate system between -1 and 1 for both axes is arbitrary.

Once the new coordinates have been obtained, they must be converted into the screen coordinates, a fundamental step to display and move the pointer on the image that the surgeon is observing. Again, this step is performed by a simple linear conversion for each axis as shown in the appendix A.3.

Thus, new screen coordinates (SC) are obtained, represented by the value of the window pixels (Fig. 3.20).

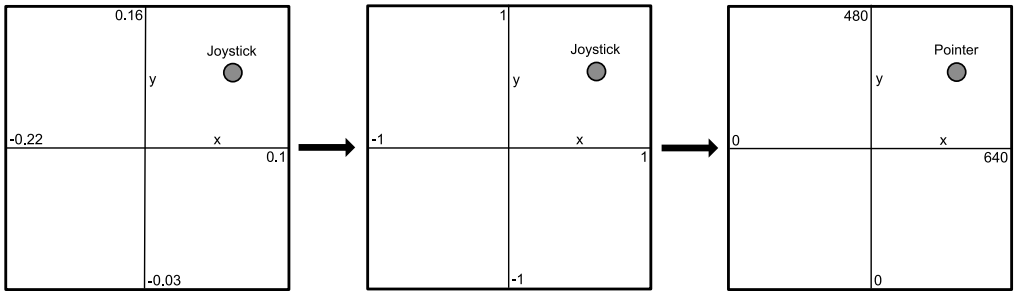


Figure 3.20: Master controller coordinates conversion

Since the pointer is a physical object defined within the world coordinate system, it is necessary to express its position in this reference system. The transition from screen coordinates to world coordinates (WC) is merely obtained by multiplying the screen coordinates with the inverse of the intrinsic

camera matrix (Eq. 3.3) previously defined:

$$WC = \begin{bmatrix} WC_x \\ WC_y \end{bmatrix} = K^{-1} \begin{bmatrix} SC_x \\ SC_y \end{bmatrix} \tag{3.4}$$

Also, in this case, the value of the pointer on the z-axis is ignored. In this way, the pointer position obtained is expressed in the world reference system, defined in the $z = 0$ plane.

Since the pointer must allow the surgeon to select a safety area that is, in perspective, above the image that he sees in the viewer, its position must be projected on an arbitrarily chosen plane located above all the objects in the scene.

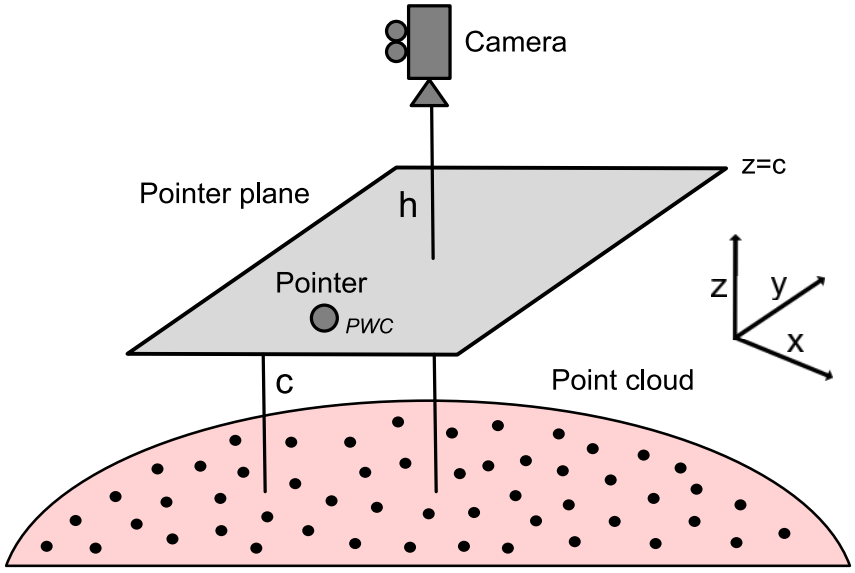


Figure 3.21: The pointer is defined on a plane above the objects of the scene.

To get the projected world coordinates (PWC), it is necessary to multiply the world coordinates of the pointer by the distance between the height of the camera on the z-axis (h) and the desired height of the pointer (c), again

on the z-axis, setting the pointer height as the z-coordinate:

$$PWC = \begin{bmatrix} PWC_x \\ PWC_y \\ PWC_z \end{bmatrix} = \begin{bmatrix} WC_x \cdot (h - c) \\ WC_y \cdot (h - c) \\ c \end{bmatrix} \tag{3.5}$$

At this point, the pointer is defined in the world reference system but projected onto the $z = c$ plane (Fig. 3.21).

The last step consists in multiplying the homogeneous coordinates just calculated with the inverse of the extrinsic matrix of the camera (T) with the z-component set to 0, so that the pointer is always in front of the camera whenever the latter changes its pose:

$$PWC = T(z = 0)^{-1} \cdot PWC = \left[\begin{array}{c|c} & x \\ R & y \\ & z = 0 \\ \hline 0 & 1 \end{array}\right]^{-1} \begin{bmatrix} PWC_x \\ PWC_y \\ PWC_z \\ 1 \end{bmatrix} \tag{3.6}$$

where R is the rotation matrix of the camera pose.

Once the pointer has been defined within the surgical scene, the surgeon can then move it using the appropriate master arm of the console. By pressing the clutch pedal, the pointer is immobilized, allowing the surgeon to position the arms more conveniently. Also, the range of motion of the pointer is limited to the rendering window edges, such as a mouse pointer of computers, which can not exit the screen even if the mouse continues to move beyond the limits of the window.

To draw the safety area, the surgeon closes the gripper on the master device with his thumb and forefinger. The system will draw a line until the surgeon releases the gripper (Fig. 3.2). If needed, the system will close the line.

Once the area is wholly drawn, it is used to reconstruct the 3D information which falls inside it, as explained in the next sections.

3.4.2 Point cloud selection

Once the safety area has been drawn, the next step is to select each point of the point cloud that is located within the area boundary from the user's point of view (Fig. 3.22).

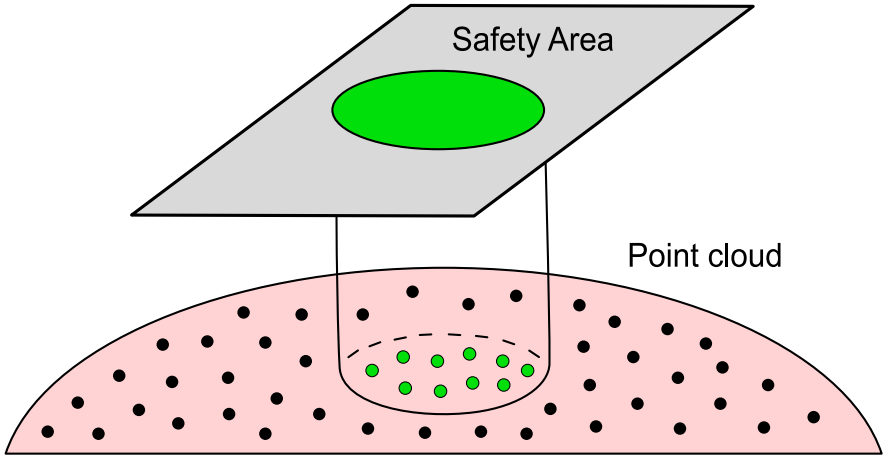


Figure 3.22: The points inside the safety area boundary are selected.

The selected points will be used to reconstruct the safety volume, from which the surgical tool will be steered away thanks to the active constraints.

The problem is defined in the 3D space, so the primary challenge consists in identifying the 3D points confined within a 2D polygon from the point of view of the user who draw the area.

Initially, attempts were made to identify points within a polyhedron using a 3D ray-casting algorithm. This polyhedron was obtained by joining the corresponding vertices of the same polygon projected on a parallel plane beyond the scene. However, this method proved to be inefficient, computationally heavy and somewhat imprecise for points on the edges of the polyhedron.

To solve the difficulties and problems deriving from the method just introduced, it was decided to transform the 3D problem into a 2D problem. Instead of projecting the safety area on a parallel plane beyond the scene to create a polyhedron, it was more convenient to project the point cloud on

the same plane of the safety area (Fig. 3.23).

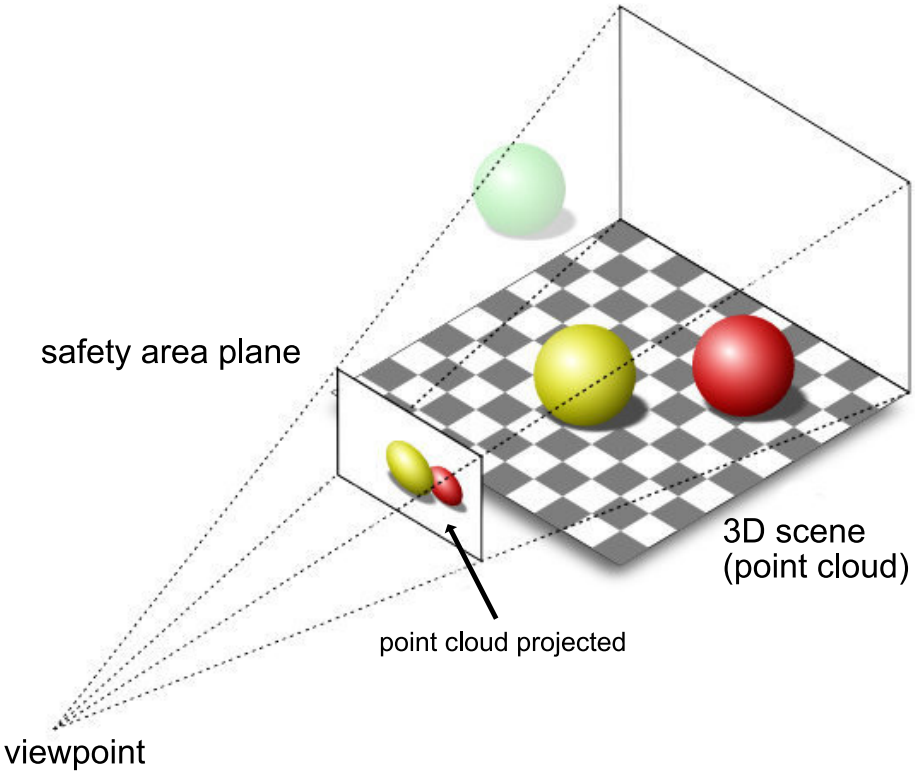


Figure 3.23: The point cloud is projected on the safety area plane [19].

Following this step, by applying a 2D ray-casting algorithm, it is possible to identify the points inside the polygon and then project them back to their original position to obtain the points of the safety volume's surface. This method, as will be shown in the next paragraphs, proved to be extremely precise, reliable and instantaneous, even for complex polygons and massive point clouds.

Perspective projection

A mapping from three dimensions onto two dimensions, as in the case of a video camera, is called perspective projection. Rays of light enter the camera through an infinitesimally small aperture and the intersection of the light rays with a given plane form the image of the object.

Some terminologies are necessary to explain this transformation (Fig. 3.24):

- the model consists of a plane (image plane) and a 3D point O (center of projection).
- the distance f between the image plane and the center of projection O is the focal length.
- the line through O and perpendicular to the image plane is the optical axis.
- the intersection of the optical axis with the image place o is called principal point or image center (note: the principal point is not always the actual center of the image).

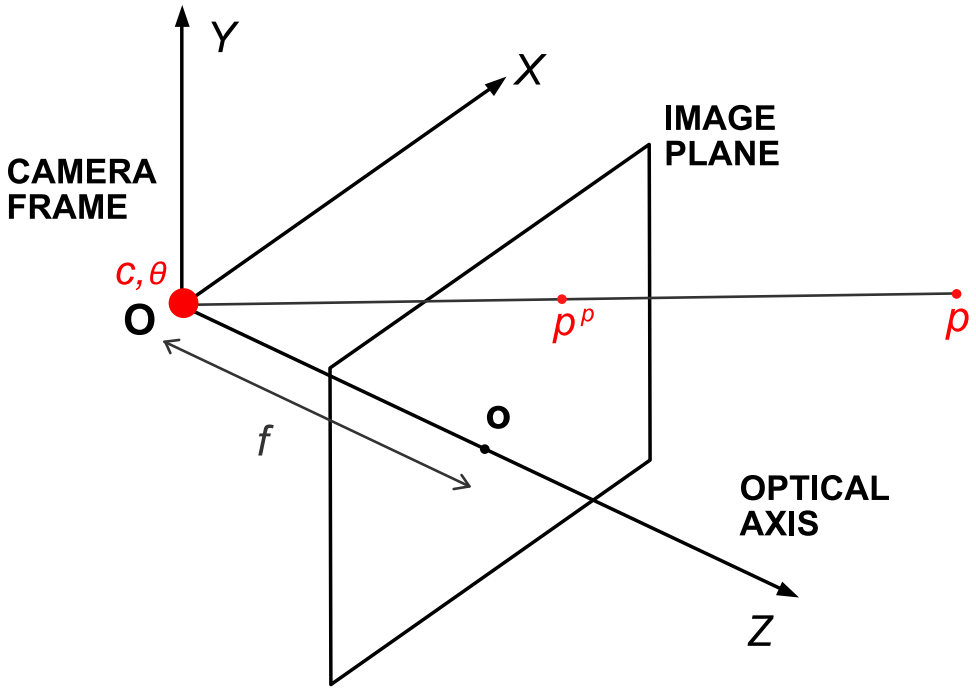


Figure 3.24: The perspective projection model [20]

The following variables are defined to describe the projection:

- $p_{x,y,z}$ - the 3D position of a point to be projected.
- $c_{x,y,z}$ - the 3D position of the camera.
- $\theta_{x,y,z}$ - the orientation of the camera.

Which results in:

- $p^p_{x,y}$ - the 2D projection of p.

In general, the world and the camera coordinate systems are not aligned. To simplify the derivation of the perspective projection equations, firstly, it is necessary to define a point $p^t_{x,y,z}$. This point represents the position of a point $p_{x,y,z}$ in the camera reference frame, with origin in $c_{x,y,z}$, and rotated by $\theta_{x,y,z}$ from the initial coordinate system. This is achieved by applying a rotation $\theta_{x,y,z}$ to the point. This transformation is often called "camera transform", and can be expressed as follows:

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin\theta_x & \cos\theta_x \end{bmatrix} \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \begin{bmatrix} \cos\theta_z & \sin\theta_z & 0 \\ -\sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.7)$$

This represents a rotation of three angles, using the xyz convention, which can be interpreted either as "rotate about the extrinsic axes (axes of the scene) in the order z, y, x (reading right-to-left)" or "rotate about the intrinsic axes (axes of the camera) in the order x, y, z (reading left-to-right)". Note that if the camera is not rotated ($\theta_{x,y,z} = \langle 0, 0, 0 \rangle$) then the matrices drop out (as identities).

The transformed point $p^t_{x,y,z}$ is therefore given by:

$$\begin{bmatrix} p^t_x \\ p^t_y \\ p^t_z \end{bmatrix} = R \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \quad (3.8)$$

This rotation is also applied to the polygon representing the safety area. In this way, the polygon is on a plane $z = h = c_z - f$ perpendicular to the z-axis, where h is the distance between the pointer and the origin (as previously defined) and therefore the distance between the plane on which the safety area is now defined and the $z = 0$ plane.

The position of the camera is also transformed:

$$\begin{bmatrix} c_x^t \\ c_y^t \\ c_z^t \end{bmatrix} = R^{-1} \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} \tag{3.9}$$

In this way, the following situation is verified:

- the camera axis (optical axis) is aligned with the world's z-axis.
- avoid image inversion by assuming that the image plane is in front of the center of projection.

To finally project the point cloud on the $z = h$ plane, the equations of the perspective projection have to be derived.

Considering the following similar triangles (Fig. 3.25),

$$\begin{aligned} \text{from } OA'B' \text{ and } OAB: \quad & \frac{f}{Z} = \frac{r}{R} \\ \text{from } A'B'C' \text{ and } ABC: \quad & \frac{x}{X} = \frac{y}{Y} = \frac{r}{R} \end{aligned} \tag{3.10}$$

the perspective equations can be obtained:

$$x = \frac{f}{Z} \cdot X \qquad y = \frac{f}{Z} \cdot Y \qquad z = f \tag{3.11}$$

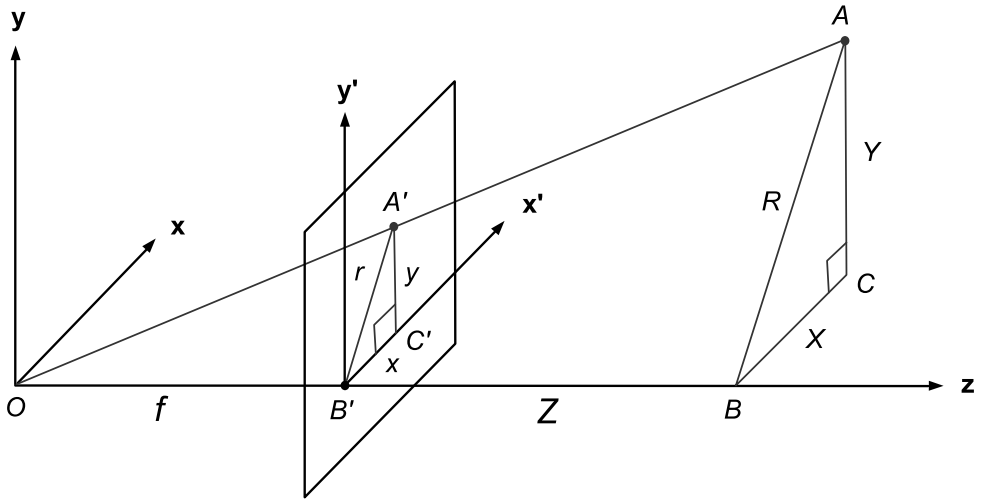


Figure 3.25: Perspective projection similar triangles [20]

By applying these equations to each rotated point, the point cloud is then projected on the plane, perpendicular to the z-axis, on which the safety area is now represented.

After having explained the theory of the perspective projection and the methods used to project the point cloud on the plane on which the safety area is defined, it is now necessary to show the process of selecting the points that lie within it. This passage is the cornerstone of this thesis work since the success of the definition of the constrained region depends on this step. The selection of points within the safety area must be exact to guarantee excellent surface reconstruction precision and virtually instantaneous to ensure the real-time factor to the entire system.

This represents the classic point-in-polygon (PIP) problem which asks whether a given point in the plane lies inside, outside, or on the boundary of a polygon. Its resolution, developed in this project, is based on the Jordan curve theorem [54], whose proof is shown in the appendix B.1.

This method identifies the included points by running semi-infinite rays out from the points, and counting how many edges they cross (Fig. 3.26).

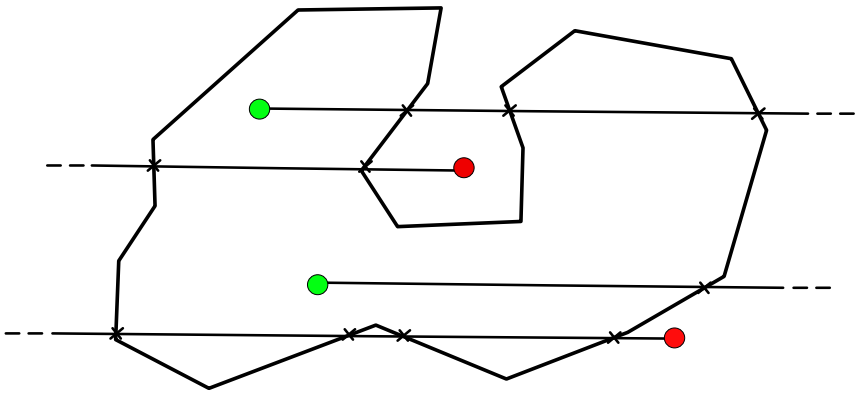


Figure 3.26: The point in polygon problem. A semi-infinite ray is sent out from the test point and the number of intersections with the polygon is evaluated.

If the number of intersections is zero or even, the point is outside the polygon. On the contrary, in case the number of intersections is odd, the point is inside the polygon.

The implementation of the theorem is based on the PNPOLY (Point Inclusion in Polygon) algorithm [55], whose code is shown in the appendix A.4.

This method runs a semi-infinite ray horizontally (increasing x, fixed y) out from the test point, and count how many edges it crosses. At each crossing, the result switches between inside and outside.

The PNPOLY method divides the plane into points inside the polygon and points outside the polygon. Points that are on the boundary are classified as either inside or outside.

Any particular point is always classified consistently the same way. In Figure 3.27, consider what PNPOLY would return when the red point, P, is tested against the two triangles, T_L and T_R. Depending on internal roundoff errors, PNPOLY may return that P is in T_L or in T_R. However, it will always give the same answer when P is tested against those triangles. That is, if

PNPOLY finds that P is in T_L, then it will find that P is not T_R. If PNPOLY finds that P is not in T_L, then it will find that P is in T_R.

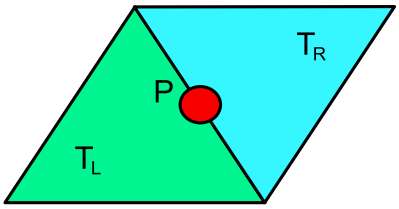


Figure 3.27: Point on boundary

This algorithm proved to be extremely robust and precise in the various trials conducted to test its effectiveness. No cases of computational errors or instances of imprecision in the selection of points within the considered polygon have ever happened. Since the system operates on massive point clouds, made up of thousands and thousands of points, the accuracy of the algorithm, albeit important, is not as fundamental as its computational speed. The safety area must be generated almost instantaneously to preserve the real-time nature of the system.

Various tests were performed to verify the computational speed of the algorithm. For each trial, a polygon was traced manually, with larger dimensions, as well as the number of vertices and sides that compose it. In this way, a point cloud of increasing size has been considered within it. The time necessary for the algorithm to identify all the points included in the polygon was evaluated. The tests were conducted on a workstation with:

- CPU Intel Core i7-6700K @ 4.0 GHz
- Number of cores: 8
- RAM: 16 Gb
- Graphics: GeForce GTX 980 Ti

The results are shown in Table 3.1.

Number of points	Number of polygon's edges	Time (ms)
30	45	93
307	79	132
1358	114	181
3125	169	232
3960	338	450
5699	523	528

Table 3.1: PNPOLY benchmark

Considering that intraoperative point clouds of about 5000 points can represent with high precision the anatomical structures under examination, the algorithm proves to be fast in identifying the points included in the polygon, both for massive point clouds and for polygons composed of a large number of sides.

3.4.3 Surface reconstruction

After identifying the points inside the safety area, it is essential to reconstruct the three-dimensional surface of the safety volume of interest, to provide the user with an explicit representation of the selected structure.

In fact, this surface, in addition to being used as a reference for the definition of active constraints, must provide the surgeon with useful visual feedback that allows him to operate with a clear view of the operative scene.

To accomplish this step, the points identified as belonging to the safety area are first re-projected in their original position within the 3D space and then exploited to reconstruct the surface.

It was chosen to implement two distinct surface reconstruction methodologies, each of which has advantages and disadvantages.

In the first place, it was decided to reconstruct a fictitious surface by merely highlighting the points belonging to the safety area with a light and

bright color (e.g., a bright green), that allowed the surgeon to easily distinguish the selected structure among the multiple anatomical structures of the operative scene (Fig. 3.28).

This method proved to be extremely fast, since the coloring process is performed simultaneously with the point selection algorithm, and therefore does not add any extra delay. The results presented Table 3.1 in the previous section remain unchanged, even if the selected points are colored.

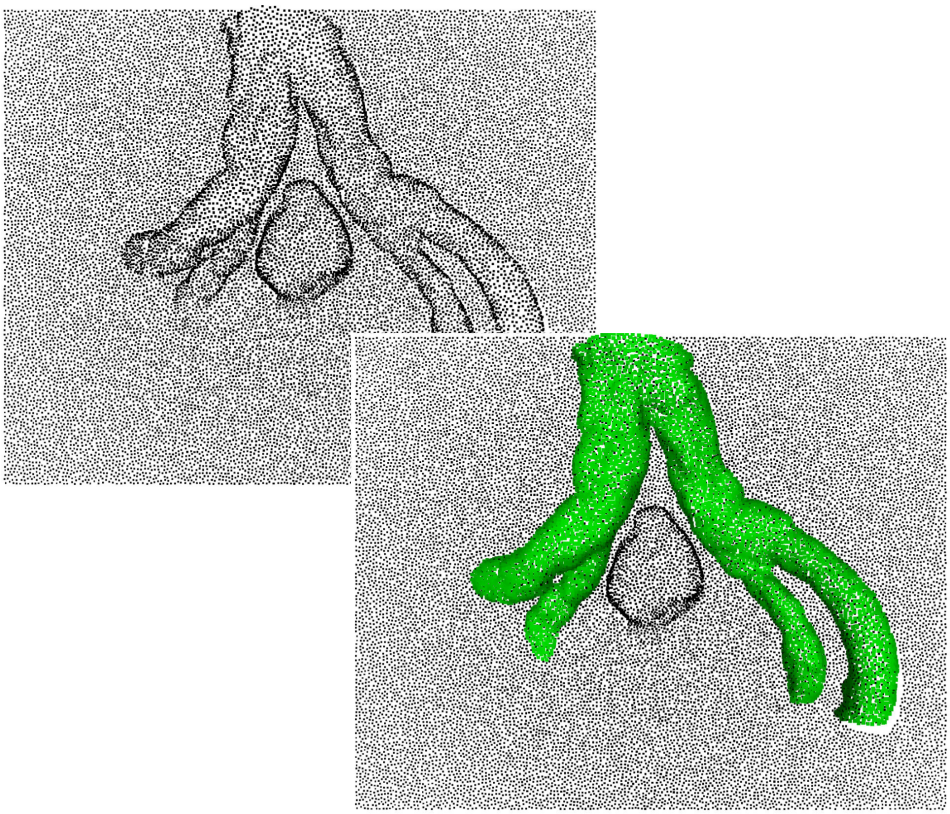


Figure 3.28: Example of surface reconstruction by coloring points

However, this fictitious surface has some disadvantages of visualization compared to a surface obtained with the classic triangulation algorithms. First of all, since it is only a representation of points in space, it does not provide a realistic shading that emphasizes shapes, curvatures, and depths. Moreover, this method requires a very dense point cloud to obtain a reason-

ably realistic representation.

The second method implemented for the reconstruction of the surface, instead, allows creating a realistic mesh starting from the selected points. This algorithm, called "Poisson Surface Reconstruction," is based on the work of Kazhdan, Bolitho, and Hoppe [56] which is reported in the appendix B.2. Its implementation, also reported in the appendix A.5, was obtained using the PCL library.

In this case, the input is represented by a dataset composed by the points and their surface normals. Starting from the set of all the normals, which are linked to the gradient of the surface, the algorithm derives the shape of the initial object by solving the Poisson problem, hence the name.

By its nature, however, this algorithm can only be used in the processing of watertight objects, i.e., closed object. This can be problematic in cases where point clouds are taken through stereoscopic cameras. In fact, the objects we are going to rebuild do not always have this property.

Specifically, the reconstruction of anatomical structures whose point cloud is derived from the inside of the patient can give unwanted results. If the final surface is not watertight, the algorithm will try to close it, often adding surfaces where there should not be. The output of this algorithm is, however, the final mesh of the given point cloud as input (Fig. 3.29).

Unlike the previous method, the Poisson reconstruction algorithm produces a real surface, whose characteristics make it much more similar to the surfaces of the real intraoperative scene. Moreover, thanks to the addition of realistic textures and computer graphics algorithms that make the surfaces deformable, the surface assumes a great realistic representation.

However, as previously mentioned, in the case of non-watertight surfaces the final reconstruction could present considerable inaccuracies that could confuse the users rather than help them. Moreover, due to the high computational complexity of the algorithm, it is not possible to obtain a precise and reliable surface in real-time.

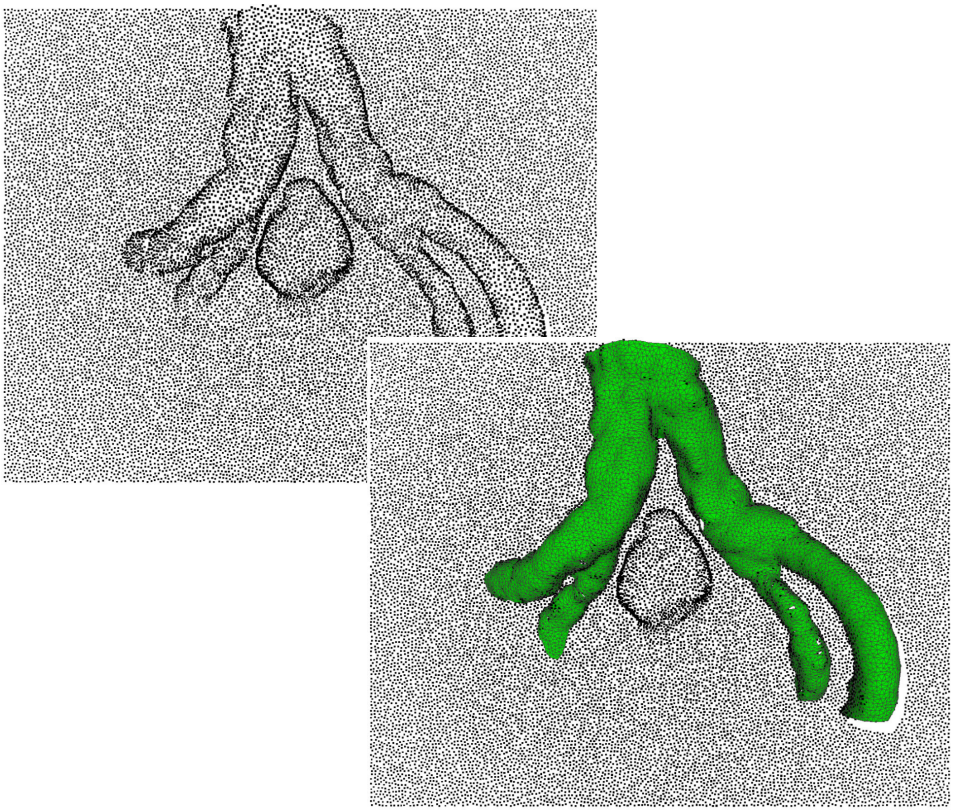


Figure 3.29: Example of the Poisson surface reconstruction

Tests were carried out to measure the required computational time of the Poisson surface reconstruction algorithm. The first method, whilst crude, does not introduce any additional delay to the time necessary to identify the points belonging to the safety area. Table 3.2 shows only the surface reconstruction times with the Poisson method. These times need to be added to the times of the previous Table 3.1 to obtain the total time due to the selection of the points and the reconstruction of the surface. The tests were conducted using the same instrumentation listed in the previous chapter and considering the same sets of points of the previous cited trials.

Number of points	Time (s)
30	0.8
307	4
1358	17
3125	43
3960	46
5699	58

Table 3.2: Poisson surface reconstruction benchmark

As previously mentioned, each of the two methods has advantages and disadvantages, which do not necessarily indicate which method is better than the other. The choice of one of the two algorithms depends entirely on the user's preferences and, in this regard, it was decided to leave the latter the choice of the reconstruction method at the start of the system.

3.4.4 Closest point identification

Following the reconstruction of the selected 3D surface, it is necessary to identify the point on the latter closest to the robotic tool (Fig. 3.30).

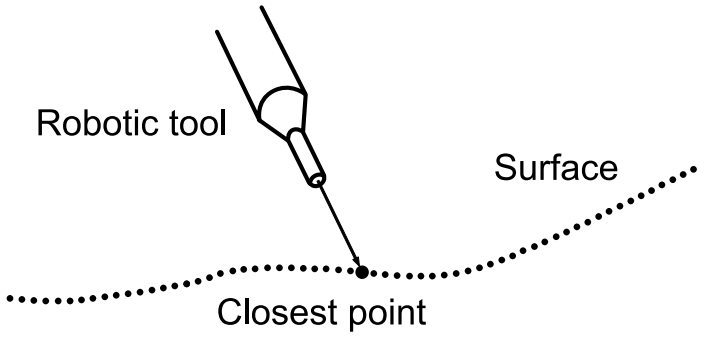


Figure 3.30: Closest point identification

This is a common problem in 3D applications, and its resolution is of fundamental importance for the calculation of the repulsive force acting on the

master arm of the surgical console, thus providing adequate tactile feedback to the user.

The search for the nearest point is carried out within a separate thread with respect to the main one that deals with the refresh of the graphics and the update of the logic related to the task. The latter, in fact, has a refresh rate of 30 Hz, clearly too low to ensure stable tactile feedback. The search functions of the point are then called inside the *HapticThread* method which, as previously introduced, represents a parallel thread that can be adjusted to a refresh rate far higher than the main one, e.g., 500-1000 Hz.

To define the problem, let D be a set of N points. D is the target dataset (e.g., the selected point cloud) and let Q be the query dataset (e.g., the tooltip). Given a query point $\mathbf{q} \in Q$ and the points from the target data set $\mathbf{x} \in D$, the goal is to find the closest point

$$NN(\mathbf{q}, D) = \arg\min_{\mathbf{x} \in D} |\mathbf{q} - \mathbf{x}|_2. \tag{3.12}$$

It should be noted that the nearest neighbor of $\mathbf{q}$ in D may not be unique, since more points in D may have the same distance from $\mathbf{q}$. However, considering rounding errors and floating point accuracy, it is sporadic for this to happen.

The target dataset D is divided into a set of Voronoi cells. The individual Voronoi cells of the Voronoi diagram of D are denoted as $voro(\mathbf{x})$, which can be seen as a closed set.

The use of the Voronoi cells represents a classic approach to solve the nearest neighbor problem.

A query point $\mathbf{q}$ is always contained in the Voronoi cell of its nearest point. So, finding the Voronoi cell that contains $\mathbf{q}$ is equivalent to finding $NN(\mathbf{q}, D)$. However, the irregularity of the Voronoi tessellation structure does not allow a direct lookup. To overcome this problem, it is, therefore, necessary to use an octree, a tree data structure in which each internal node has exactly eight children. Octrees are most often used to partition a three-dimensional space by recursively subdividing it into eight octants. This allows creating a more

regular structure on the top of the Voronoi diagram, which allows to find the corresponding Voronoi cell quickly.

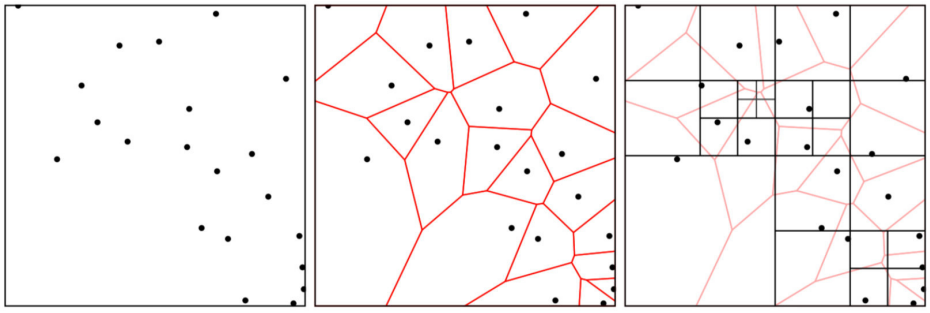


Figure 3.31: Example in 2D of the hierarchical voxel structure. The image shows, starting from the left, the raw point cloud, the Voronoi subdivision and the octree subdivision [21].

Each voxel is splitted based on the number of intersecting Voronoi cells: each voxel that intersects more than $\mathbf{M}_{max}$ Voronoi cells is split into eight sub-voxels, which are processed recursively (Fig. 3.31). The resulting set of data points whose Voronoi cells intersect a voxel v is denoted

$$L(D, v) = \{\mathbf{x} \in D : voro(\mathbf{x}) \cap v \neq 0\}. \tag{3.13}$$

Therefore, for each query point $\mathbf{q}$ contained in the voxel v_{leaf}, the Voronoi cell of the closest point $NN(\mathbf{q}, D)$ must intersect v_{leaf}.

The result of the recursive subdivision is an octree. To find the nearest point of a query point $\mathbf{q}$ two steps are then required: find the leaf voxel v_{leaf} containing $\mathbf{q}$ and search for all points in $L(D, v_{\text{leaf}}(\mathbf{q})$ for the closest point of $\mathbf{q}$ [57].

The solution to this problem was achieved thanks to the *vtkCellLocator* class which belongs to the VTK library.

vtkCellLocator is a spatial search object to quickly locate cells in 3D. This class uses a uniform-level octree subdivision, where each octant carries an indication of whether it is empty or not, and each leaf octant carries a

list of the cells inside of it (an octant is not empty if it has one or more cells inside of it).

The member function *vtkCellLocator::FindClosestPoint*, of the above mentioned class, returns the closest target point $\mathbf{x}$ to the query point $\mathbf{q}$ and the Euclidean distance between them.

The closest point to the robotic tool is thus identified at the refresh rate of the *HapticThread* function and used for the definition of the active constraints, as shown at the beginning of the Chapter 3.

Chapter 4

Experimental studies and results

The primary aim of this section is to evaluate the performances of the developed system. A virtual environment interface was developed for the experimentation, which will be described in the next paragraphs as well as the experimental procedure and the results.

4.1 Experimental setup

The experimentation set comprised the virtual environment interface and the da Vinci Research Kit (dVRK, provided by Intuitive Surgical to Politecnico di Milano) (Fig. 4.1).

To prevent the computational load of the 3D graphics from affecting the haptic control loop performance, the tests were executed on a desktop computer with:

- CPU: Intel Core i7-6700K @ 4.0 GHz
- Number of cores: 8
- RAM: 16 Gb
- Graphics: GeForce GTX 980 Ti

Figure 4.1: The console of the da Vinci Research Kit

The experiment represented a simulation of a partial nephrectomy procedure. Within the virtual 3D scene, a model of a renal tumor and a model of renal arteries were arranged, both placed on a surface representing the kidney (Fig. 4.2). To give a realistic representation to the scene and, at the same time, to improve the feeling of depth for the user, the models were represented with realistic textures.

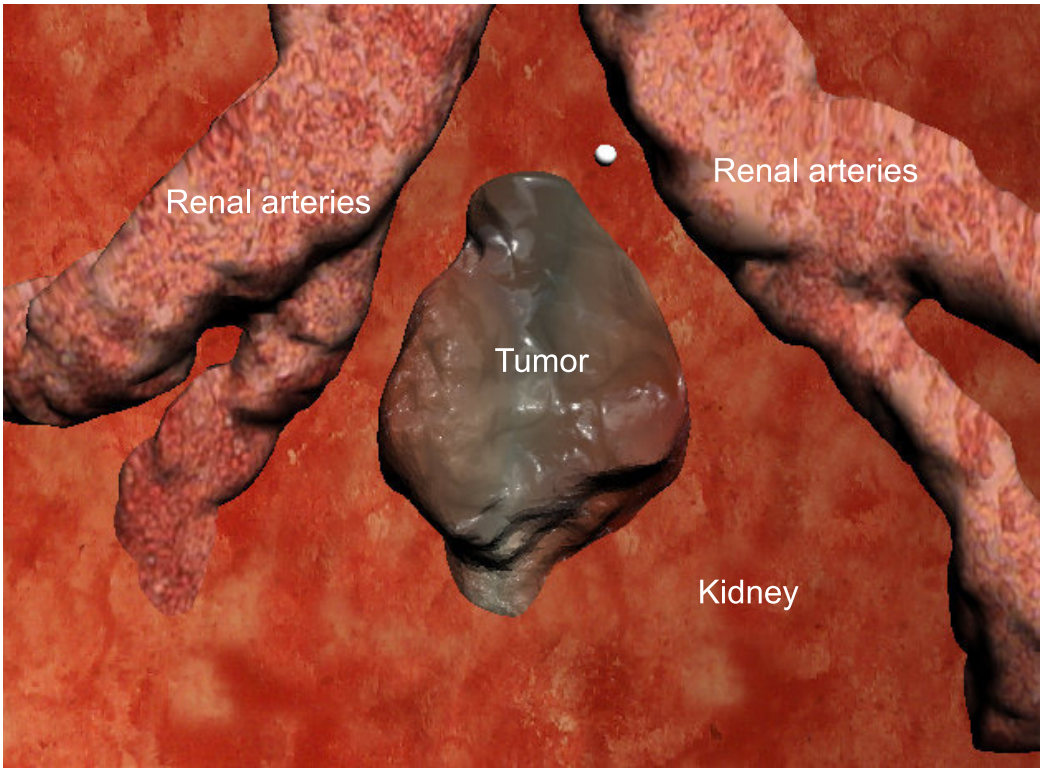


Figure 4.2: The virtual scene

For each candidate, as shown in the following sections, it was requested to perform a simulated tumor removal in two different cases. In one case the candidate just had to remove the tumor without having to trace the safety area, and therefore without the aid of any force feedback and other visual aids. In the second case, the candidate had to draw the safety area and remove the tumor with the assistance of the force feedback, generated by the safety constraints, and additional visual feedbacks.

In the safety area trial, the candidate was asked to draw an area that

enclosed the renal arteries, trying to follow, with the utmost precision, the anatomical contours of these. The pointer, as already explained in the previous chapter, is guided by moving the left or right master arm of the da Vinci console. By holding down the gripper, the user can then draw the contour of the area (Fig. 4.3).

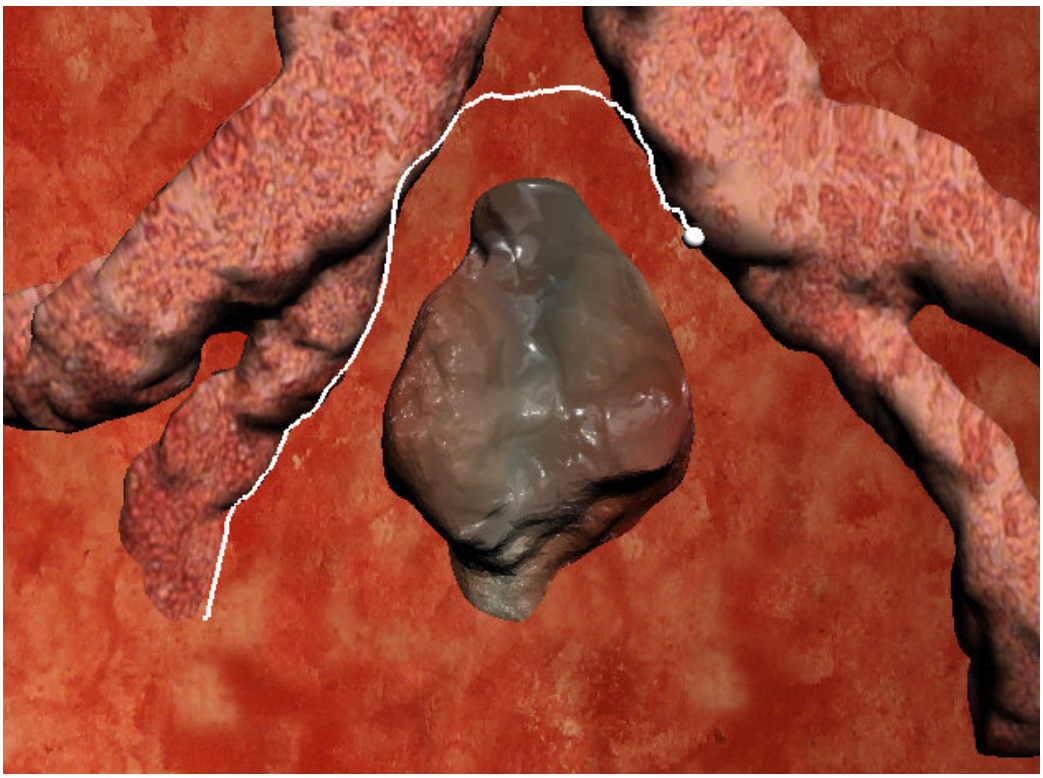


Figure 4.3: Safety area drawing

Once satisfied with the drawn contour, the gripper can be released, and the contour is automatically closed by joining the starting point with the terminal point. The safety volume is then reconstructed by the algorithm, and the virtual arms of the da Vinci are shown on the screen (Fig. 4.4). It was decided to represent the points of the area with a bright green, thus avoiding to use the Poisson method previously introduced for the reasons related to the real-time factor as shown in Chapter 3.

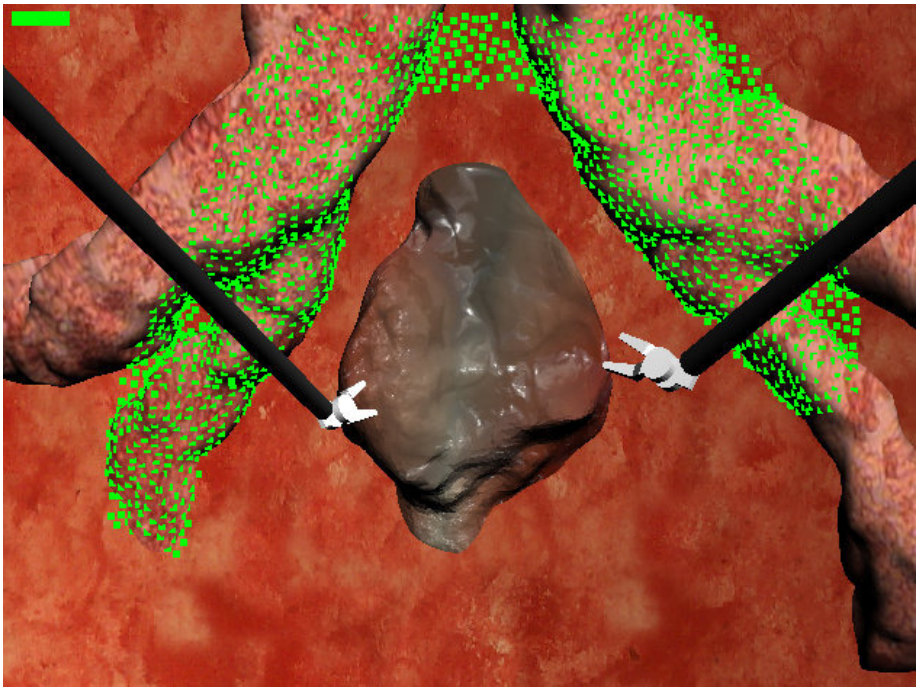


Figure 4.4: Surface reconstruction of the safety volume

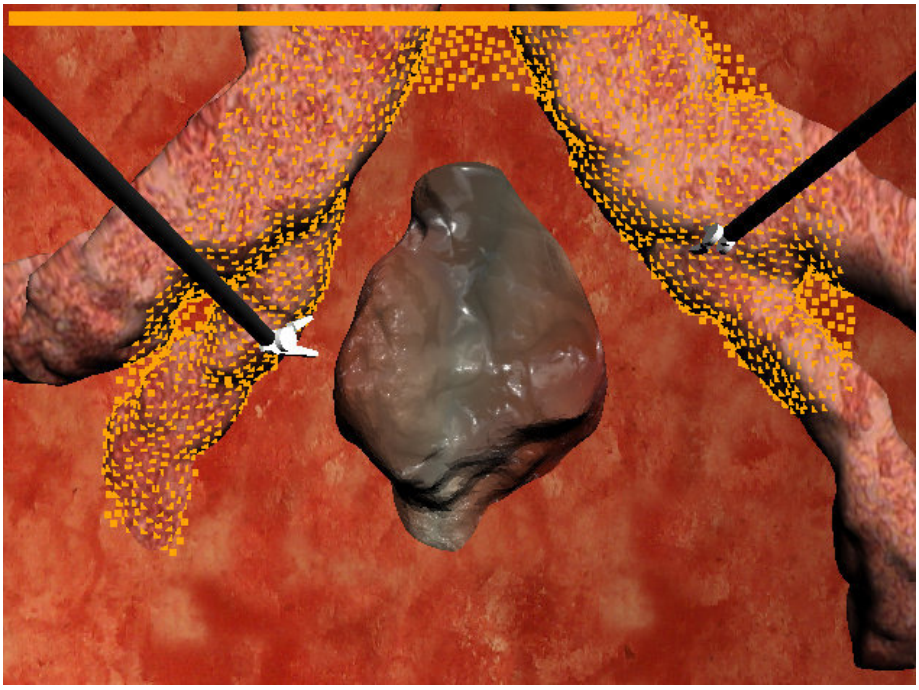


Figure 4.5: Visual aids

To give further visual feedback to the user, a colored mapping was implemented to the reconstructed area: by approaching the robotic tool to the safety volume, its surface changes color, going from a bright green to an intense red. Also, a bar, positioned at the top of the screen, changes length and color accordingly, always to indicate the proximity of the robotic tool to the surface (Fig. 4.5).

In both cases, with or without the safety area, to simulate the cut inflicted by the robotic tool on the surface, a blue line is shown on the screen, representing the torn surface by the user (Fig. 4.6). This is performed by pressing the gripper of the relative master arm. Also, the tool turns green to help the user understand when the robotic tool is actually in contact with the surface.

The test is considered completed once the simulated cut surrounds the tumor.

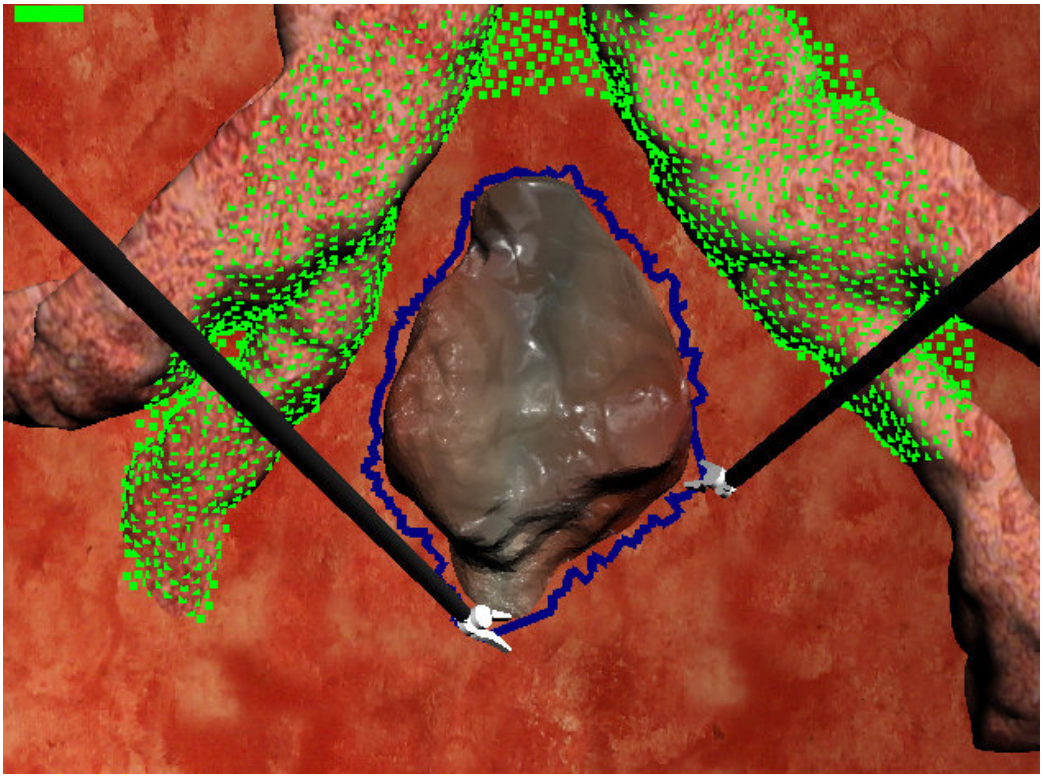


Figure 4.6: Simulated removal of the tumor

4.2 Acquisition protocol

To estimate the objective performance of the system in terms of accuracy, the following metrics have been recorded for each candidate and each trial:

- the number of collisions between the end-effector and the renal arteries
- the total duration T of the individual collisions T_i $\quad (T = \sum_i T_i)$
- the percentage of healthy tissue removed respect to the tumor surface

In particular, the last metric was calculated as

$$\%\ of\ healthy\ tissue\ removed = \frac{(A_s - A_t)}{A_t} \cdot 100 \tag{4.1}$$

where A_s represents the area of the torn surface, and A_t represents the tumor surface (Fig. 4.7).

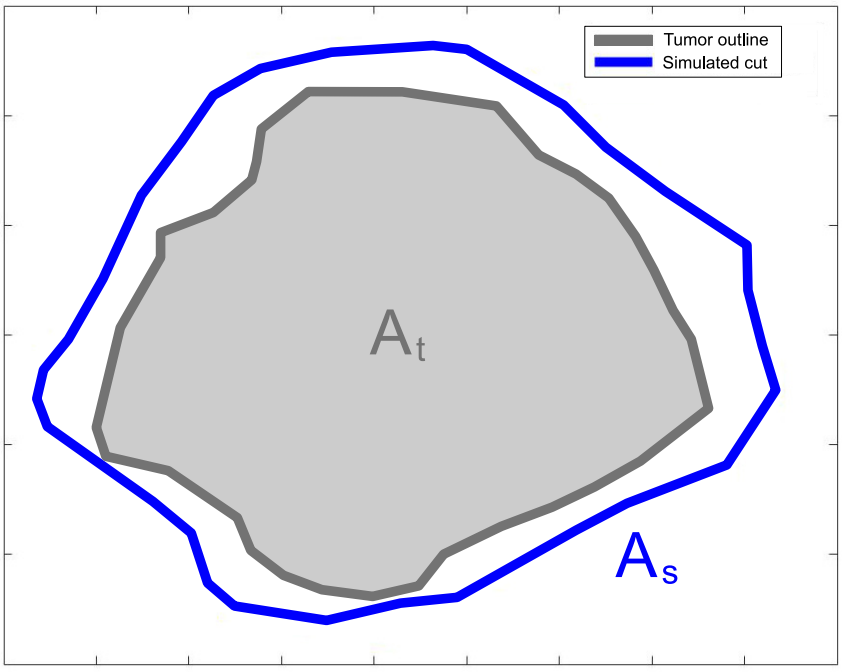


Figure 4.7: Areas of the tumor and the tissue removed by the user

The percentage of non-removed tumor tissue was not recorded, as each candidate completely removed the tumor in every trial.

In order to test the research hypothesis, a user study was carried out with volunteer users, each of which has previously signed an informed consent. The study population consisted of 10 non-medical participants (aged between 22 and 28 years old, six males and four females, nine right-handed and one left-handed) with none to little experience with robotic teleoperation.

Each volunteer was asked to perform both the unconstrained task, that is the only removal of the tumor without the visual and force feedbacks, and the constrained task, tracing the safety area first and removing the tumor with the consequent force feedback and visual aids.

Firstly, the subjects were introduced to the dVRK console, the experimental protocol and the final evaluation modalities, as well as they were shown videos of successful execution of the task.

Then the users were given as much time as they needed to train with the console. The average of training time per user was 13 minutes (minimum of 8 minutes and maximum of 15 minutes).

Finally, all the subjects went through the final test, consisting of three repetitions of the constrained task and three repetitions of the unconstrained task. The parameters of each repetition were averaged for each type of task.

The linear elastic coefficient k_p was set to 1000 $\frac{N}{m}$, and the linear damping coefficient k_d was set to 10 $\frac{Ns}{m}$. The maximum repulsive force was set to 4 N. Each parameter has been empirically calibrated so as to guarantee excellent stability and accuracy to the force feedback generated.

4.3 Results analysis

Due to the small sample size, non-parametric statistical significance tests were used to compare the performance of the users doing the constrained and unconstrained task. This assumption was, however, confirmed by the result of the Lilliefor test, performed in MATLAB to check the non-normal distribution of parameters ($p < 0.001$ for each metric). The Wilcoxon rank

sum test was then employed in MATLAB to determine the statistical significance of the sets.

Statistically significant effects were assessed at $p < 0.05$.

The distributions of the number of collisions, the total collisions time and the percentage of healthy tissue removed for the constrained and unconstrained task are visualized respectively in Figures 4.8, 4.9 and 4.10 and their median values are reported in Table 4.1.

	Unconstrained	Constrained	p
Number of collisions	1.7500	0.5000	< 0.001
Total duration of collisions (s)	4.2015	0.7835	0.0013
% of healthy tissue removed	29.09	20.25	< 0.001

Table 4.1: Parameter median values

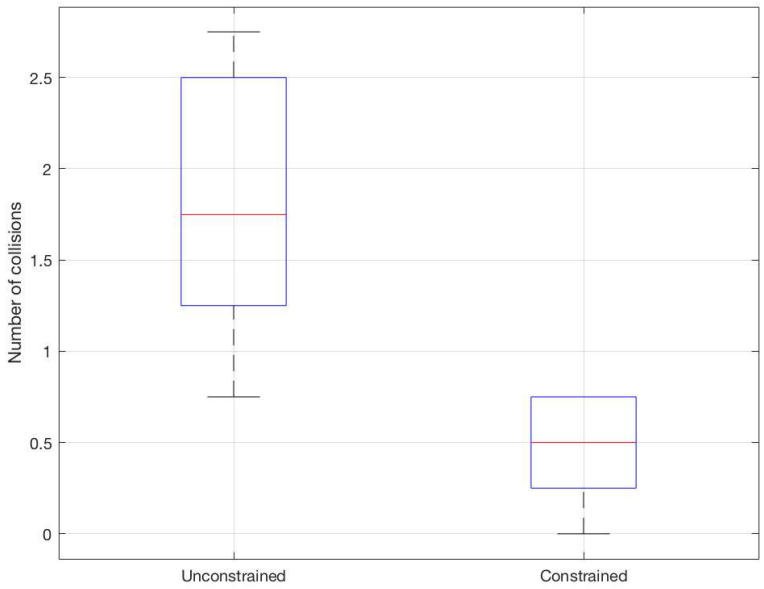


Figure 4.8: Number of collisions

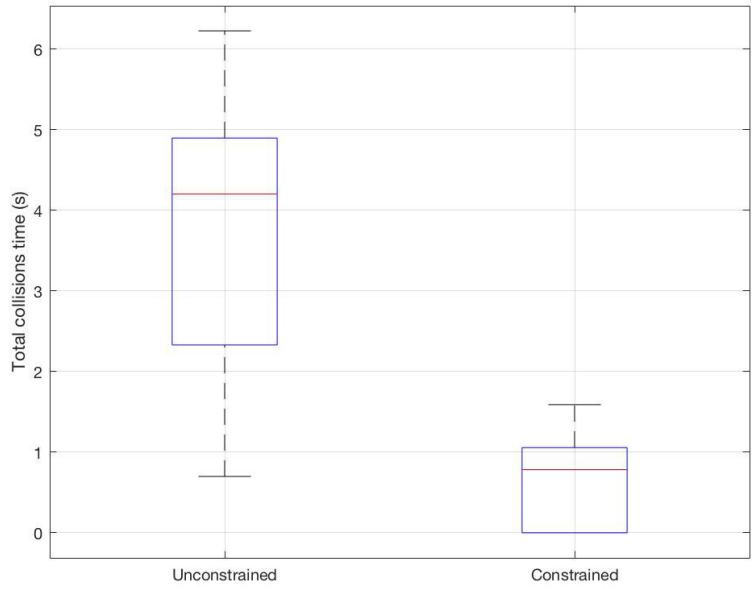


Figure 4.9: Total duration of collisions

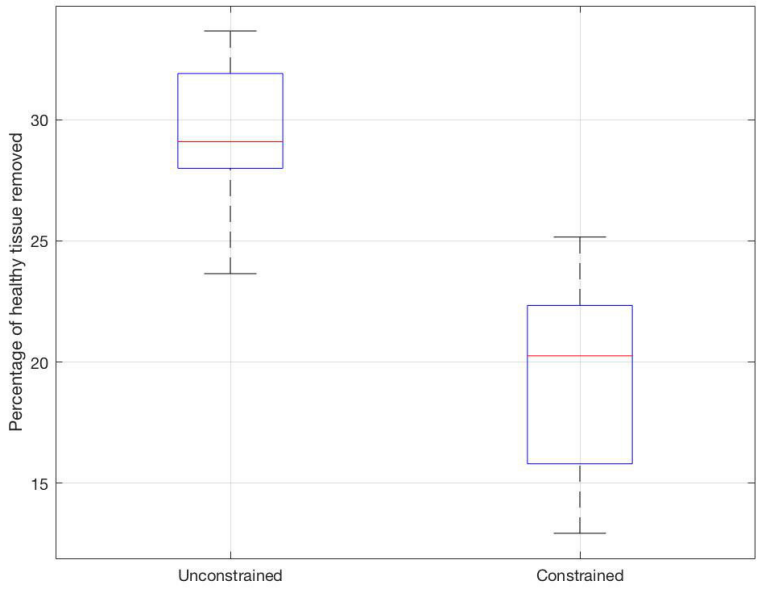


Figure 4.10: Percentage of healthy tissue removed

The results suggest that the safety constraints, defined from the safety volume, have enhanced the accuracy in completing the task compared to the unconstrained case.

Considering the number of collisions and the total duration of these collisions, the constrained method demonstrated statistically significant improvements compared to the unconstrained case, reducing both the parameters.

Furthermore, safety constraints have reduced the variability of the performance among the users compared to the unconstrained sets. The variability of the number of collisions and the duration of the collisions is a direct result of the different user skills. During the experiments, it was observed that some subjects had difficulty in performing the tasks efficiently due to their lack of experience with the surgical console. Since the safety constraints impart a repulsive force on the master arms, the subject could easily be guided by them in the narrow passages between the tumor and the renal arteries.

This fact is also confirmed by the reduced percentage of healthy tissue removed in the constrained case compared to the unconstrained case. In fact, the repulsive forces produced by the surface of the safety volume, in the restricted areas between the tumor and the renal arteries, require the user to make the cut near the tumor, thus reducing the excess tissue removed.

This is clearly visible also in the Figure 4.11, which represents the contour of the tumor and the path executed by the user to remove the latter, both in the constrained case and in the unconstrained case.

From the figure, it is possible to notice that, in the constrained case, the path executed by the user follows more faithfully the contour of the tumor where the latter is extremely near to the renal arteries (highlighted by the red zones), while in the other areas the path is substantially at the same distance from the tumor. This can not be affirmed for the unconstrained case, in which the path realized by the user is maintained more or less at the same distance from the tumor, even in the narrower passages between the latter and the arteries.

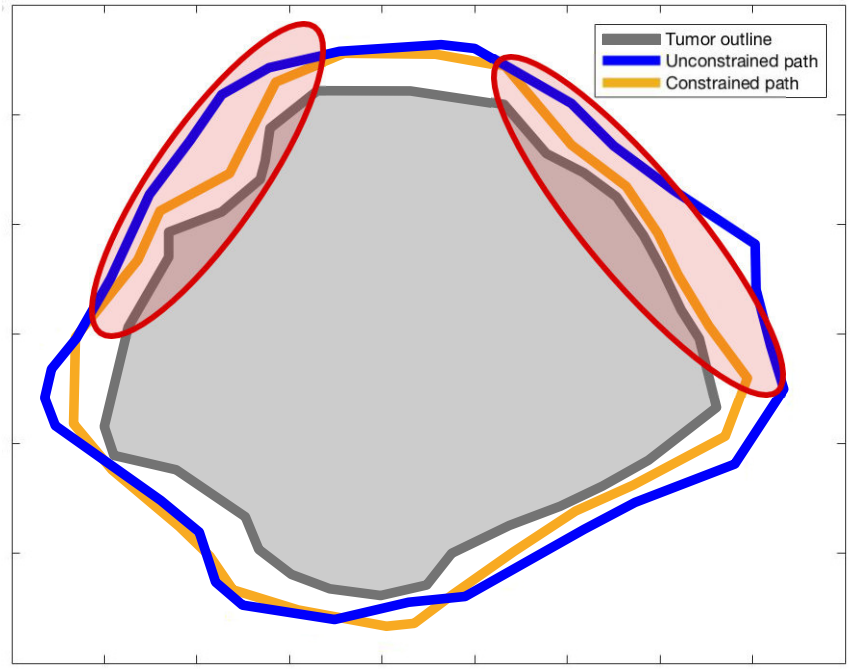


Figure 4.11: Example of the tumor's outline and the user cutting paths

Finally, the average task times (considering only the tumor removal time) presented in Table 4.2 and in Figure 4.12, show that users have completed the constrained task faster, that can implicitly imply ease of use and reduction of cognitive load.

	Unconstrained	Constrained
Average time (s)	40.7951	32.6393

Table 4.2: Average tumor removal time values

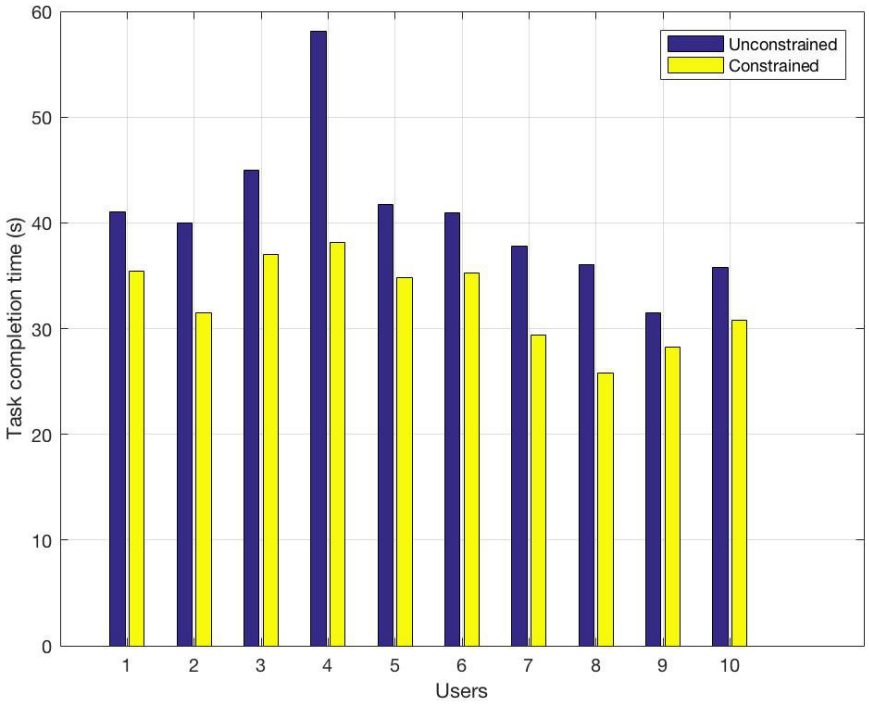


Figure 4.12: Tumor removal times

To conclude, the overall results show that the introduced viscosity-based constraints, defined from the surface of the safety volume, can achieve accuracy enhancements compared to the traditional unconstrained case.

Chapter 5

Conclusions and future work

The current thesis work, achieved at the Neuroengineering and Medical Robotics Laboratory (Nearlab), department of "Elettronica, Informazione e Bioingegneria" (DEIB), aims at developing a novel intraoperative system for the robot-assisted surgery field, in order to improve its current limitations.

This chapter concludes this dissertation by providing a summary of the results exposed in the preceding section and suggestions of areas for future research.

5.1 Summary of contributions

The system attempts to overcome the current limits of robotic surgery, which, on the one hand, represents a highly innovative field of surgery but, on the other hand, it is still in the early stage of its technological development and then it does not entirely contain the human error.

In the field of robotic surgery the risk of damaging fragile and critical regions, such as vessels, arteries, and nerves, is still high and the management of the main risk factors lies entirely in the dexterity of the surgeon.

The system designed, therefore, tries to overcome these limits by giving the user the possibility to define a safety area, which, thanks to the surface reconstruction algorithms, allows to define the active constraints, thus generating repulsive forces that can remove the robotic tool from the previously

mentioned critical regions.

The operational flow of the system starts with the definition of the safety area. It is drawn through a pointer shown on the screen, directly freehand by the surgeon and through the same joysticks used to teleoperate the robot. The definition of a safety area drawn intraoperatively allows solving the problems found in the previous methods of registration of preoperative models on the surgical scene. In fact, these methods are not able to predict the dynamic changes that may occur between the time of acquisition of preoperative information and the effective start of the surgical procedure.

The points of the point cloud, enclosed in the safety area, are used for the reconstruction of the safety volume. This volume is then exploited to define the active constraints, necessary to generate the repulsive forces that move the robotic tool away from the selected anatomical structure.

To determine the usefulness and accuracy of the system, tests were carried out on ten volunteer candidates. They performed a particular task of removing a renal tumor near the renal arteries, trying to avoid the collision between the latter and the robotic tools. Candidates had to perform the task in the absence of the force feedback and then repeat it by first drawing the safety area and then removing the tumor with the help of the forces generated by the active constraints. The results were promising, showing that the performances of the constrained task were significantly higher.

5.2 Avenues for future research

The results obtained were encouraging. In particular, the system conceived has made possible to improve the pre-existing methods, overcoming some of the limitations that they presented. However, the research focused only on a small group of candidates.

For a better statistical determinism, it would be advisable to carry out multiple session experiments involving a larger sample size.

Additional analyzes should also involve a further acquisition group made up of experts in robotic teleoperation and a realistic augmented reality task

in order to evaluate the effectiveness of the system in the real world.

Furthermore, it would be necessary to implement different types of constraints (i.e., not only visco-elastic regional constraints), in order to compare them and evaluate which one could be more efficient in the execution of the task

Finally, the current work will be integrated into a much larger project called "SMARTsurg" (SMart weArable Robotic Teleoperated surgery), which has received funding from the "European Union's Horizon 2020 research and innovation programme" and is currently under development by a team of highly experienced clinical, academic, and industrial partners across the Europe, included the Politecnico di Milano (Fig. 5.1).

Figure 5.1: SMARTsurg logo and partners [22]

The project will develop an advanced system for performing robot assisted minimally invasive surgery to reduce the surgeon's cognitive load related to the system's operation to shorten training time and deliver accuracy, safety, reduced procedure time and expanded applicability.

Advanced features will be developed and integrated into the proposed platform including dexterous anthropomorphic surgical instruments, wearable hand exoskeletons with haptic feedback to control the surgical instruments, wearable smart glasses for augmented reality and 3D reconstruction

of the surgical field and the system introduced in this work for defining intraoperative safety constraints [59] (Fig. 5.2).

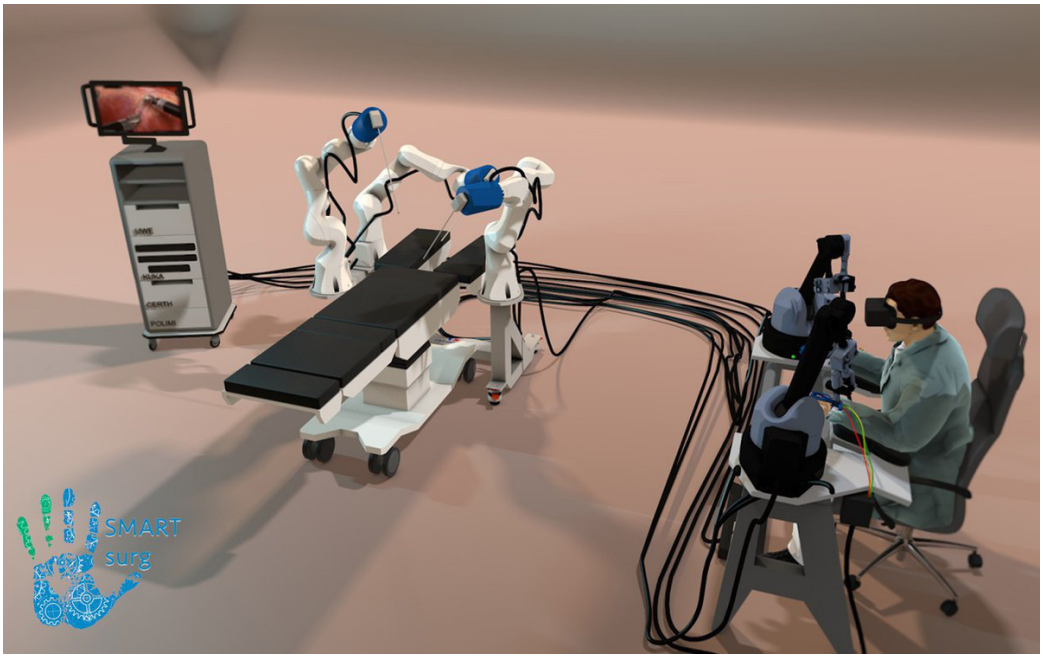


Figure 5.2: The SMARTsurg system [22]

To conclude, it is hoped that the work presented in this thesis will provide a strong starting point for future research in the field of new surgical technologies and robotic surgery.

Appendix A

Code

A.1 Rendering windows configuration

```
bool ar_mode = false;
int n_views = 3;
bool one_window_per_view = false;
bool borders_off  = true;
std::vector<int> view_resolution = {640, 480};
std::vector<int> window_positions = {1280, 0};

graphics = new Rendering(view_resolution, ar_mode, n_views,
    one_window_per_view, borders_off, window_positions);
```

where:

- *'ar_mode'* allows creating an augmented reality task (by setting the background images as the images received from the camera) or a virtual reality task.
- *'n_views'* sets the number of the graphical views generated.
- *'one_window_per_view'* allows to set the number of the windows created (i.e., one unique window for the all views or one window for each view).

– *'borders_off'* sets the borders of the window by making the latter movable with the computer mouse or fixed.

– *'view_resolution'* represents the graphical window size (e.g., 640 pixel width and 480 pixel height for the da Vinci screens).

– *'windows_position'* sets the position in pixel of the window on the computer desktop space and the da Vinci screen space.

A.2 Joystick pose conversion

```
// The first value of the arrays is referred to the x-axis, the second value to the y-axis.
double OldMin[2] = {-0.22, -0.03};
double OldMax[2] = {0.1, 0.06};
double OldRange[2];

int NewMin = -1;
int NewMax = 1;
int NewRange;

double OldValue[2];
double NewValue[2];

OldRange[0] = OldMax[0] - OldMin[0];
OldRange[1] = OldMax[1] - OldMin[1];

NewRange = NewMax - NewMin;

NewValue[0] = (((OldValue[0] - OldMin[0]) * NewRange) / OldRange[0]) + NewMin;
NewValue[1] = (((OldValue[1] - OldMin[1]) * NewRange) / OldRange[1]) + NewMin;
```

where:

- *'OldMin'* and *'OldMax'* represent the da Vinci joystick workspace limits (e.g., -0.22 and 0.1 on the x-axis and -0.03 and 0.06 on the y-axis for the left master arm).
- *'NewMin'* and *'NewMax'* represent the new coordinate system limits (-1 and 1 for both axes).
- *'OldValue'* and *'NewValue'* represent the value of the master arm in the da Vinci coordinate system and the transformed value in the new symmetrical coordinate system respectively.

A.3 Screen coordinates conversion

```
int WindowSize[2] = {480, 640};

double OldValue[2];
double NewValue[2];

NewValue[0] = 0.5 * (OldValue[0] + 1) * WindowSize[0];
NewValue[1] = 0.5 * (OldValue[1] + 1) * WindowSize[1];
```

where:

- *'WindowSize'* represents the window dimensions.
- *'OldValue'* represents the master arm position in the previous defined coordinate system.
- *'NewValue'* represents the position defined in the screen coordinate system.

A.4 Point Inclusion in Polygon

```
int pnpoly(int nvert, float *vertx, float *verty, float testx,
 ↪  float testy)
{
  int i, j, c = 0;
  for (i = 0, j = nvert-1; i < nvert; j = i++) {
    if ( ((verty[i]>testy) != (verty[j]>testy)) &&
         (testx < (vertx[j]-vertx[i]) * (testy-verty[i]) /
          ↪  (verty[j]-verty[i]) + vertx[i]) )
       c = !c;
  }
  return c;
}
```

where:

- *'nvert'* is the number of vertices in the polygon.
- *'vertx'* and *'verty'* are the arrays containing the x and y-coordinates of the polygon's vertices.
- *'testx'* and *'testy'* are x and y-coordinates of the test point.

A.5 Poisson Surface Reconstruction

```
vtkSmartPointer<vtkPolyData>
 ↪  SimPointCloud::PoissonReconstruction(
    vtkSmartPointer<vtkPolyData> polydata) {

    PointCloud<PointXYZ>::Ptr cloud(new PointCloud<PointXYZ>);

    // Convert the dataset
    io::vtkPolyDataToPointCloud(polydata, *cloud);

```

```
9       // Begin the passthrough filter
10      PointCloud<PointXYZ>::Ptr filtered(new
        ↪  PointCloud<PointXYZ>());
11      PassThrough<PointXYZ> filter;
12      filter.setInputCloud(cloud);
13      filter.filter(*filtered);
14      cout << "Passthrough filter complete" << endl;
15
16      if (cloud->isOrganized()) {
17          // Integral images normals estimation
18          IntegralImageNormalEstimation<PointXYZ, Normal> ne;
19          ne.setInputCloud(filtered);
20          ne.setNormalEstimationMethod(
21              IntegralImageNormalEstimation<PointXYZ,
                ↪  Normal>::COVARIANCE_MATRIX);
22          ne.setNormalSmoothingSize(float (0.5));
23          ne.setDepthDependentSmoothing(true);
24          ne.compute(normals);
25
26      } else {
27          // OMP normals estimation
28          NormalEstimationOMP<PointXYZ, Normal> ne;
29          ne.setNumberOfThreads(8);
30          ne.setInputCloud(filtered);
31          ne.setKSearch(3);
32          Eigen::Vector4f centroid;
33          compute3DCentroid(*cloud, centroid);
34          ne.setViewPoint(centroid[0], centroid[1], centroid[2]);
35          ne.compute(normals);
36
37          // Reverse normals' direction
38          for (size_t i = 0; i < normals.size(); ++i) {
```

```
39              normals.points[i].normal_x *= -1;
40              normals.points[i].normal_y *= -1;
41              normals.points[i].normal_z *= -1;
42          }
43      }
44
45      // Combine points and normals
46      PointCloud<PointNormal>::Ptr cloud_smoothed_normals(new
        ↪  PointCloud<PointNormal>());
47      concatenateFields(*filtered, normals,
        ↪  *cloud_smoothed_normals);
48
49      // Begin Poisson reconstruction
50      Poisson<PointNormal> poisson;
51      poisson.setDepth(9);
52      poisson.setInputCloud(cloud_smoothed_normals);
53      PolygonMesh mesh;
54      poisson.reconstruct(mesh);
55
56      // Convert the mesh
57      vtkSmartPointer<vtkPolyData> vtk_mesh =
        ↪  vtkSmartPointer<vtkPolyData>::New();
58      VTKUtils::convertToVTK(mesh, vtk_mesh);
59
60      return vtk_mesh;
61  }
```

This function, at first, filters the points to eliminate any outliers and then checks if the point cloud is organized or not. An organized point cloud dataset is the name given to point clouds that look like an organized structure of images (or matrices), in which the data is broken down into rows and columns. The advantages of an organized data set are that knowing the relationship between adjacent points, the nearest neighboring operations are

much more efficient, thus accelerating the calculation and reducing the costs of certain algorithms in PCL. If the point cloud is organized, the surface normals are estimated using the "Integral Images Normal Estimation" method. Otherwise, this is made using the "OMP (Open Multi-core Paradigms) Normal Estimation" method, which speeds up the computation to compensate for the fact that the point cloud is not organized. Finally, the points and the surface normals are combined, and through the function for the reconstruction of the surface with the Poisson method, the final mesh of the point cloud is obtained.

Appendix B

General notions

B.1 The Jordan curve theorem for polygons

The Jordan curve theorem states that any simple closed curve C divides the points of the plane not on C into two distinct domains (with no points in common) of which C is the common boundary.

It is necessary to give a proof of this theorem for the case where C is a closed polygon P, as the safety area is.

In this regard, the proof of the theorem from Courant and Robbins [54] is reported.

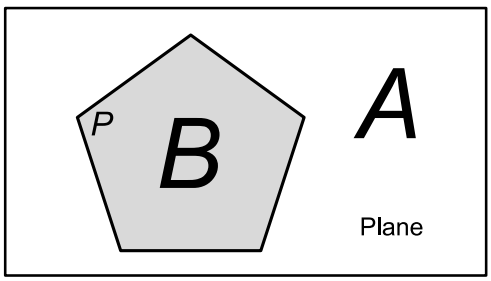


Figure B.1: The Jordan curve theorem statement

The points of the plane not on P fall into two classes, A and B, such that any two points of the same class can be joined by a polygonal path which does not cross P, while any path joining a point of A to a point of B must

cross P. The class A will form the "outside" of the polygon, while the class B will form the "inside" (Fig. B.1).

To proof the theorem, a fixed direction in the plane can be chosen. This direction is not parallel to any of the sides of P and, since P has but a finite number of sides, this is always possible (Fig. B.2). The classes A and B are defined as follows:

1. The point p belongs to A if the ray through p in the fixed direction intersects P in an even number, 0, 2, 4, 6, ..., of points.

2. The point p belongs to B if the ray through p in the fixed direction intersects P in an odd number, 1, 3, 5, ..., of points.

With regard to rays that intersect P at vertices, we shall not count an intersection at a vertex where both edges of P meeting at the vertex are on the same side of the ray, but we shall count an intersection at a vertex where the two edges are on opposite sides of the ray. We shall say that two points p and q have the same "parity" if they belong to the same class, A or B.

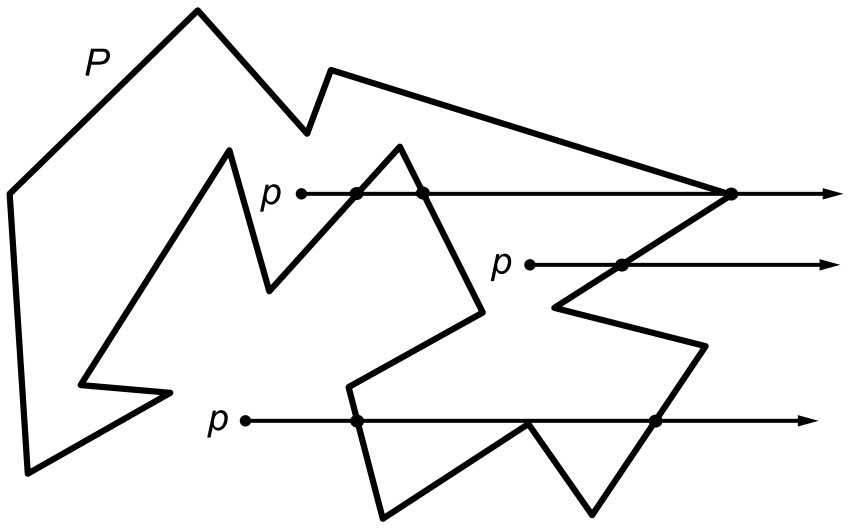


Figure B.2: Semi-rays intersections

First, it is important to notice that all the points on any line segment not intersecting P have the same parity. For the parity of a point p moving

along such a segment can only change when the ray in the fixed direction through p passes through a vertex of P, and in neither of the two possible cases will the parity actually change, because of the agreement made in the preceding paragraph. From this, it follows that if any point p_1 of A is joined to a point p_2 of B by a polygonal path, then this path must intersect P, for otherwise the parity of all the points of the path, and in particular of p_1 and p_2, would be the same. Moreover, any two points of the same class, A or B, can be joined by a polygonal path which does not intersect P. Call the two points p and q. If the straight segment pq joining p to q does not intersect P it is the desired path. Otherwise, let p' be the first point of intersection of this segment with P, and let q' be the last such point (Fig. B.3).

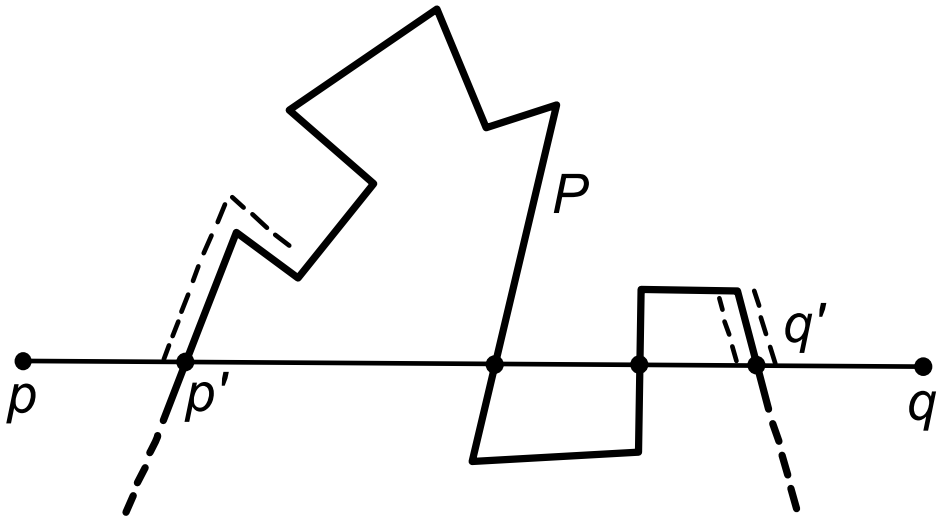


Figure B.3: Counting intersections

Construct the path starting from p along the segment pp', then turning off just before p' and following along P until P returns to pq at q'. If this path can't be proved to intersect pq between q' and q, rather than between p' and q', then the path may be continued to q along $q'q$ without intersecting P. It is clear that any two points r and s near enough to each other, but on opposite sides of some segment of P, must have different parity, for the ray through r will intersect P in one more point than will the ray through

s. Thus it is clear that the parity changes as we cross the point q' along the segment pq. It follows that the dotted path crosses pq between q' and q, since p and q (and hence every point on the dotted path) have the same parity.

This completes the proof of the Jordan curve theorem for the case of a polygon P. The "outside" of P may now be identified as the class A, since if we travel far enough along any ray in the fixed direction we shall come to a point beyond which there will be no intersection with P, so that all such points have parity 0, and hence belong to A. This leaves the "inside" of P identified with the class B. No matter how twisted the simple closed polygon P, we can always determine whether a given point p of the plane is inside or outside P by drawing a ray and counting the number of intersections of the ray with P. If this number is odd, then the point p is imprisoned within P, and cannot escape without crossing P at some point. If the number is even, then the point p is outside P.

B.2 The Poisson Surface Reconstruction

The Poisson surface reconstruction method, from Hoppe *et al.* [56], exploits an implicit function framework. Specifically, it computes a 3D indicator function χ (defined as one at points inside the model, and zero at points outside), and then obtain the reconstructed surface by extracting an appropriate iso-surface. The key insight is represented by the integral relationship between the indicator function of a model and the oriented points sampled from the surface of the latter. Also, the gradient of the indicator function is a vector field that is zero almost everywhere. In fact, except at points near the surface, where it is equal to the inward surface normal, the indicator function is constant almost everywhere. The oriented point samples can be then viewed as samples of the gradient of the model's indicator function (Fig. B.4).

The indicator function computation thus reduces to inverting the gradient operator. It consists in finding the scalar function χ whose gradient best approximates a vector field $\vec{V}$ defined by the samples, i.e. $\min_\chi \|\nabla\chi - \vec{V}\|$. By applying the divergence operator, this variational problem transforms into

a standard Poisson problem: compute the scalar function χ whose Laplacian (divergence of the gradient) equals the divergence of the vector field $\vec{V}$,

$$\Delta\chi \equiv \nabla \cdot \nabla\chi = \nabla \cdot \vec{V}. \tag{B.1}$$

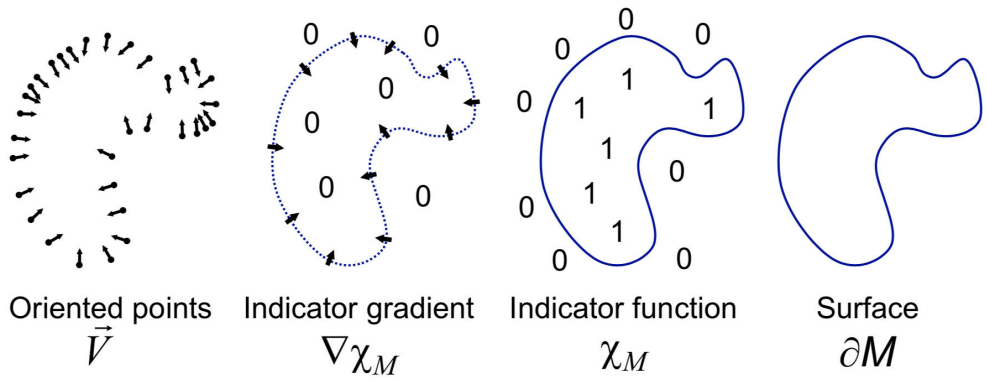


Figure B.4: Illustration of the Poisson reconstruction in 2D [23]

The input data S is represented by a set of samples $s \in S$, each consisting of a point $s.p$ and an inward-facing normal $s.\vec{N}$, assumed to lie on or near the surface ∂M of an unknown model M. The goal is then to reconstruct a watertight, triangulated approximation to the surface, realized by approximating the indicator function of the model and finally extracting the isosurface.

The indicator function is a piecewise constant function, then the explicit computation of its gradient field would result in a vector field with unbounded values at the surface boundary. This can be avoided by convolving the indicator function with a smoothing filter. The following lemma formalizes the relationship between the gradient of the indicator function and the normal surface field.

Lemma: Given a solid M with boundary ∂M, let χ_M denote the indicator function of M, $\vec{N}_{\partial M}(p)$ be the inward surface normal at $p \in \partial M$, $\tilde{F}(q)$ be a smoothing filter, and $\tilde{F}_p(q) = \tilde{F}(q-p)$ its translation to the point p. The gradient of the smoothed indicator function is equal to the vector field

obtained by smoothing the surface normal field:

$$\nabla(\chi_M * \tilde{F})(q_0) = \int_{\partial M} \tilde{F}_q(q_0)\vec{N}_{\partial M}(p)dp. \tag{B.2}$$

Proof: To prove this, equality for each of the components of the vector field can be showed. Computing the partial derivative of the smoothed indicator function with respect to x:

$$\begin{aligned}
\left.\frac{\partial}{\partial x}\right|_{q_0}(\chi_M * \tilde{F}) &= \left.\frac{\partial}{\partial x}\right|_{q=q_0}\int_M \tilde{F}(q-p)dp \\
&= \int_M (-\frac{\partial}{\partial x}\tilde{F}(q_0-p))dp \\
&= -\int_M \nabla\cdot(\tilde{F}(q_0-p),0,0)dp \\
&= \int_{\partial M}\Big\langle(\tilde{F}_p(q_0,0,0),\vec{N}_{\partial M}(p)\Big\rangle dp.
\end{aligned} \tag{B.3}$$

The first equality follows from the fact that χ_M is equal to zero outside of M and one inside. The second follows from the fact that $(\partial/\partial q)\tilde{F}(q-p) = -(\partial/\partial p)\tilde{F}(q-p)$. The last follows from the Divergence Theorem. A similar argument shows that the y and z components of the two sides are equal, thereby completing the proof.

The input set of oriented points provides precisely enough information to approximate the integral with a discrete summation. Specifically, using the point set S to partition ∂M into distinct patches $\mathscr{P}_s \subset \partial M$, the integral can be approximated over a patch $\mathscr{P}_s$ by the value at point sample $s.p$, scaled by the area of the patch:

$$\begin{aligned}
\nabla(\chi_M * \tilde{F})(q) &= \sum_{s\in S}\int_{\mathscr{P}_s}\tilde{F}_p(q)\vec{N}_{\partial M}(p)dp \\
&\approx \sum_{s\in S}|\mathscr{P}_s|\tilde{F}_{s.p}(q)s.\vec{N} \equiv \vec{V}(q).
\end{aligned} \tag{B.4}$$

It should be noted that though Equation B.2 is true for any smoothing filter $\tilde{F}$, in practice, care must be taken in choosing the filter. In particular, the filter should satisfy two essential conditions. It should be sufficiently narrow to not over-smooth the data, and it should be wide enough so that the integral over $\mathscr{P}_s$ is well approximated by the value at $s.p$ scaled by the patch area. An excellent choice of filter that balances these two requirements is the classic Gaussian, whose variance is on the order of the sampling resolution.

Having formed a vector field $\vec{V}$, it is necessary to solve for the function $\tilde{\chi}$ such that $\nabla\tilde{\chi} = \vec{V}$. However, $\vec{V}$ is generally not integrable (i.e., it is not curl-free), so an exact solution does not generally exist. To find the best least-squares approximate solution, the divergence operator can be applied, to finally form the Poisson equation

$$\Delta\tilde{\chi} = \nabla \cdot \vec{V}. \tag{B.5}$$

Implementation

This reconstruction algorithm is made under the assumption that the point samples are uniformly distributed over the model surface. At first, it is necessary to define a space of functions with high resolution near the surface of the model and coarser resolution away from it. After that, the next steps consist into expressing the vector field $\vec{V}$ as a linear sum of functions in previously mentioned space, setting up and solving the Poisson equation, and finally extracting an isosurface of the resulting indicator function.

First, it is mandatory to choose the space of functions in which to discretize the problem. The choice of a regular 3D grid represents the simplest approach, but such a uniform structure becomes impractical for fine-detail reconstruction. In fact, the dimension of the space is cubic in the resolution while the number of surface triangles grows quadratically.

An accurate representation of the implicit function is only necessary near the reconstructed surface. This last assumption motivates the use of an adaptive octree both to represent the implicit function and to solve the Poisson system. Specifically, the positions of the sample points can be used to

define an octree $\mathscr{O}$ and associate a function F_o to each node $o \in \mathscr{O}$ of the tree, choosing the tree and the functions so that the following conditions are satisfied:

- The vector field $\vec{V}$ can be precisely and efficiently represented as the linear sum of the F_o.
- The matrix representation of the Poisson equation, expressed in terms of the F_o can be solved efficiently.
- A representation of the indicator function as the sum of the F_o can be precisely and efficiently evaluated near the surface of the model.

Given a maximum three depth D and a set of point samples S, the octree $\mathscr{O}$ can be defined to be the minimal octree with the property that every point samples falls into a leaf node at depth D.

Next, it is necessary to define a space of functions obtained as the span of translates and scales of a fixed, unit-integral, base function $F : \mathbb{R}^3 \to \mathbb{R}$. For every node $o \in \mathscr{O}$, F_o is set to be the unit-integral "node function" centered about the node o and stretched by the size of o:

$$F_o(q) \equiv F\left(\frac{q - o.c}{o.w}\right)\frac{1}{o.w^3}. \tag{B.6}$$

where $o.c$ and $o.w$ are the center and width of node o.

This space of functions $\mathscr{F}_{\mathscr{O},F} \equiv \text{Span}\{F_o\}$ has a multiresolution structure similar to that of traditional wavelet representations. Finer nodes are associated with higher-frequency functions, and the function representation becomes more precise as we near the surface.

In selecting a base function F, the goal is to choose a function so that the vector field $\vec{V}$, defined in Equation B.3, can be precisely and efficiently represented as the linear sum of the node functions $\{F_o\}$.

If the position of each sample would be replaced with the center of the leaf node containing it, the vector field $\vec{V}$ could be efficiently expressed as

the linear sum of $\{F_o\}$ by setting:

$$F(q) = \tilde{F}\left(\frac{q}{2^D}\right). \tag{B.7}$$

This way, each sample would contribute a single term (the normal vector) to the coefficient corresponding to its leaf's node function. Since the sampling width is 2^{-D} and the samples all fall into leaf nodes of depth D, the error arising from the clamping can never be too big (at most, on the order of half the sampling width).

Finally, since a maximum tree depth of D corresponds to a sampling width of 2^{-D}, the smoothing filter should approximate a Gaussian with variance on the order of 2^{-D}. Thus, F should approximate a Gaussian with unit-variance.

For efficiency, it's better to approximate the unit-variance Gaussian by a compactly supported function so that the resulting Divergence and Laplacian operators are sparse, and the evaluation of a function expressed as the linear sum of F_o at some point q only requires summing over the nodes $o \in \mathscr{O}$ that are close to q. Thus, setting F to be the n-th convolution of a box filter with itself, this results in the base function F:

$$F(x,y,z) \equiv (B(x)B(y)B(z))^{*n} \quad \text{with} \quad B(t) = \begin{cases} 1 & |t| < 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{B.8}$$

To allow for sub-node precision, it is better to avoid clamping a sample's position to the center of the containing leaf node and instead use trilinear interpolation to distribute the sample across the eight nearest nodes. Thus, the approximation to the gradient field of the indicator function can be defined as:

$$\vec{V}(q) \equiv \sum_{s \in S} \sum_{o \in \mathrm{Ngbr}_D(s)} \alpha_{o,s} F_o(q) s.\vec{N} \tag{B.9}$$

where $\mathrm{Ngbr}_D(s)$ are the eight depth-D nodes closest to $s.p$ and $\{\alpha_{o,s}\}$ are the trilinear interpolation weights.

Since the samples are uniform, it can be assumed that the area of a patch $\mathscr{P}_s$ is constant and $\vec{V}$ is a good approximation, up to a multiplicative constant, of the gradient of the smoothed indicator function.

Poisson solution

Having defined the vector field $\vec{V}$, the next step consists into solving for the function $\tilde{\chi} \in \mathscr{F}_{\mathscr{O},F}$ such that the gradient of $\tilde{\chi}$ is closest to $\vec{V}$, i.e. a solution to the Poisson equation $\Delta\tilde{\chi} = \nabla \cdot \vec{V}$.

One challenge of solving for $\tilde{\chi}$ is that though $\tilde{\chi}$ and the coordinate functions of $\vec{V}$ are in the space $\mathscr{F}_{\mathscr{O},F}$ it is not necessarily the case that the functions $\Delta\tilde{\chi}$ and $\nabla \cdot \vec{V}$ are.

To address this issue, it is mandatory to solve for the function $\tilde{\chi}$ such that the projection of $\Delta\tilde{\chi}$ onto the space $\mathscr{F}_{\mathscr{O},F}$ is closest to the projection of $\nabla \cdot \vec{V}$. Since, in general, the functions F_o do not form an orthonormal basis, solving this problem directly is expensive. However, the problem can be simplified by solving for the function $\tilde{\chi}$ minimizing:

$$\sum_{o \in \mathscr{O}} \left\| \langle \Delta\tilde{\chi} - \nabla \cdot \vec{V}, F_o \rangle \right\|^2 = \sum_{o \in \mathscr{O}} \left\| \langle \Delta\tilde{\chi}, F_o \rangle - \langle \nabla \cdot \vec{V}, F_o \rangle \right\|^2. \tag{B.10}$$

Thus given the $|\mathscr{O}|$-dimensional vector v whose o-th coordinate is $v_o = \langle \nabla \cdot \vec{V}, F_o \rangle$, the goal is to solve for the function $\tilde{\chi}$ such that the vector obtained by projecting the Laplacian of $\tilde{\chi}$ onto each of the F_o is as close to v as possible.

To express this in matrix form, let $\tilde{\chi} = \sum_o x_o F_o$, so $x \in \mathbb{R}^{|\mathscr{O}|}$. Then, $|\mathscr{O}| \times |\mathscr{O}|$ matrix L is defined, such that Lx returns the dot product of the Laplacian with each of the F_o. Specifically, for all o, $o' \in \mathscr{O}$, the (o, o')-th entry of L is set to:

$$L_{o,o'} \equiv \left\langle \frac{\partial^2 F_o}{\partial x^2}, F_{o'} \right\rangle + \left\langle \frac{\partial^2 F_o}{\partial y^2}, F_{o'} \right\rangle + \left\langle \frac{\partial^2 F_o}{\partial z^2}, F_{o'} \right\rangle. \tag{B.11}$$

Thus, solving for $\tilde{\chi}$ amounts to find

$$\min_{x \in \mathbb{R}^{|\mathcal{O}|}} \|Lx - v\|^2 . \tag{B.12}$$

Note that the matrix L is sparse and symmetric.

Isosurface Extraction

To obtain a reconstructed surface $\partial\tilde{M}$, it is necessary first to select an isovalue and, from the computed indicator function, extract the corresponding isosurface.

The isovalue can be selected so that the extracted surface closely approximates the positions of the input samples. This is done by evaluating $\tilde{\chi}$ at the sample positions and use the average of the values for isosurface extraction:

$$\partial\tilde{M} \equiv \{q \in \mathbb{R}^3 \mid \tilde{\chi} = \gamma\} \quad \text{with} \quad \gamma = \frac{1}{|S|} \sum_{s \in S} \tilde{\chi}(s.p). \tag{B.13}$$

This choice of isovalue has the property that scaling $\tilde{\chi}$ does not change the isosurface. Thus, knowing the vector field $\vec{V}$ up to a multiplicative constant provides sufficient information for reconstructing the surface.

Finally, to extract the isosurface from the indicator function, a computer graphics algorithm for extracting polygonal meshes can be exploited, such as the Marching Cubes method [58].

Bibliography

[1] Maya M. Hammoud et al. «To the point: medical education review of the role of simulators in surgical training». In: *American Journal of Obstetrics and Gynecology* 199.4 (2008), pp. 338–343. ISSN: 0002-9378.

[2] Isabelle Opitz et al. «Bleeding remains a major complication during laparoscopic surgery: analysis of the SALTS database». In: *Langenbeck's Archives of Surgery* 390 (2004), pp. 128–133.

[3] Shinji Onda et al. «Identification of inferior pancreaticoduodenal artery during pancreaticoduodenectomy using augmented reality-based navigation system». In: *Journal of hepato-biliary-pancreatic sciences* 21 (Apr. 2014).

[4] Danail Stoyanov. «Surgical vision». In: *Annals of biomedical engineering* 40.2 (Feb. 2012), pp. 332–345. ISSN: 0090-6964.

[5] Veronica Penza et al. «Safety Enhancement Framework for Robotic Minimally Invasive Surgery». In: *The Hamlyn Symposium on Medical Robotics* (June 2017).

[6] Vic Velanovich. «Laparoscopic vs open surgery: a preliminary comparison of quality-of-life outcomes». In: *Surgical endoscopy* 14 (Feb. 2000), pp. 16–21.

[7] Christianne Buskens et al. «The potential benefits and disadvantages of laparoscopic surgery for ulcerative colitis: A review of current evidence». In: *Best practice & research. Clinical gastroenterology* 28 (Feb. 2014), pp. 19–27.

[8] M. A. Bruhat et al. «The benefits and risks of laparoscopic surgery». In: *Revue française de gynécologie et d'obstétrique* 88 (Mar. 1993), pp. 84–88.

[9] Anthony R. Lanfranco et al. «Robotic Surgery: A Current Perspective». In: *Annals of surgery* 239 (Feb. 2004), pp. 14–21.

[10] Michael J. Mack. «Minimally Invasive and Robotic Surgery». In: *JAMA: the journal of the American Medical Association* 285 (Mar. 2001), pp. 568–572.

[11] Allison Okamura. «Haptic Feedback in Robot-Assisted Minimally Invasive Surgery». In: *Current opinion in urology* 19 (Feb. 2009), pp. 102–107.

[12] Rahul Gupta and Abhishek Chandna. «Robotics in Surgery - Past, Present and Future». In: *Advances in Robotics and Automation* 01 (Nov. 2015).

[13] Sijo Parekattil and Michael Moran. «Robotic instrumentation: Evolution and microsurgical applications». In: *Indian journal of urology : IJU : journal of the Urological Society of India* 26 (July 2010), pp. 395–403.

[14] Steffen Rausch, C. Schmitt, and T. Kälble. «Radical Prostatectomy: An Option for High-Risk Prostate Cancer». In: *Advances in urology* 2012 (Jan. 2012).

[15] Sean G. Vesey et al. «UK radical prostatectomy outcomes and surgeon case volume: Based on an analysis of the British Association of Urological Surgeons Complex Operations Database». In: *BJU international* 109 (July 2011), pp. 346–354.

[16] Byung Ha Chung. «The role of radical prostatectomy in high-risk prostate cancer». In: *Prostate International* 1.3 (2013), pp. 95–101. ISSN: 2287-8882.

[17] Julia Finkelstein et al. «Open Versus Laparoscopic Versus Robot-Assisted Laparoscopic Prostatectomy: The European and US Experience». In: *Reviews in urology* 12 (Jan. 2010), pp. 35–43.

[18] Fatih Atug Hasan Hüseyin Tavukçu Omer Aytac. «Nerve-sparing techniques and results in robot-assisted radical prostatectomy». In: *Investigative and clinical urology* 57.Suppl 2 (Dec. 2016), S172–S184. ISSN: 2466-0493.

[19] Kimberley Hoyland et al. «Post-Radical Prostatectomy Incontinence: Etiology and Prevention». In: *Reviews in urology* 16 (Dec. 2014), pp. 181–1888.

[20] Andrew R. McCullough. «Sexual dysfunction after radical prostatectomy». In: *Reviews in urology* 7 Suppl 2 (2005), S3–S10. ISSN: 1523-6161.

[21] Andrea Salonia et al. «Prevention and Management of Postprostatectomy Sexual Dysfunctions Part 2: Recovery and Preservation of Erectile Function, Sexual Desire, and Orgasmic Function». In: *European Urology* 62.2 (2012), pp. 273–286. ISSN: 0302-2838.

[22] Lepor Herbert. «A Review of Surgical Techniques for Radical Prostatectomy». In: *Reviews in urology* 7 (2005), S11–S17.

[23] Christian Barré. «Open Radical Retropubic Prostatectomy». In: *European Urology* 52.1 (2007), pp. 71–80.

[24] Gelbert Luiz Chamon do Carmo Amorim et al. «Comparative analysis of radical prostatectomy techniques using perineal or suprapubic approach in the treatment of localized prostate cancer.» In: *Einstein* 82 (2010), pp. 200–204.

[25] Joel B. Nelson. «Debate: Open radical prostatectomy vs. laparoscopic vs. robotic». In: *Urologic oncology* 25 (Nov. 2007), pp. 490–493.

[26] S. S. Chang et al T. M. Webster S. D. Herrell. «Robotic assisted laparoscopic radical prostatectomy versus retropubic radical prostatectomy: a prospective assessment of postoperative pain». In: *The Journal of Urology* 174 (2005), pp. 912–914.

[27] Trinity J. Bivalacqua, Phillip M. Pierorazio, and Li-Ming Su. «Open, laparoscopic and robotic radical prostatectomy: Optimizing the surgical approach». In: *Surgical Oncology* 18.3 (2009). Prostate Cancer: Clinical Outcomes in Clinical Practice, pp. 233–241. ISSN: 0960-7404.

[28] T. E. Ahlering. «Robotic versus laparoscopic radical prostatectomy». In: *Nature clinical practice* 1.2 (2004), pp. 58–59.

[29] Rodrigo Frota et al. «Comparison of Radical Prostatectomy Techniques: Open, Laparoscopic and Robotic Assisted». In: *Official journal of the Brazilian Society of Urology* 34 (June 2008), pp. 259–268.

[30] J. L. Bennett J. K. Parsons. «Outcomes of retropubic, laparoscopic, and robotic-assisted prostatectomy». In: *Urology* 72.2 (Aug. 2008), pp. 412–416.

[31] Stephen A. Poon et al. «Trends in Partial and Radical Nephrectomy: An Analysis of Case Logs from Certifying Urologists». In: *The Journal of Urology* 190.2 (2013), pp. 464–469. ISSN: 0022-5347.

[32] William T. Lowrance et al. «Complications After Radical and Partial Nephrectomy as a Function of Age». In: *The Journal of Urology* 183.5 (2010), pp. 1725–1730. ISSN: 0022-5347.

[33] Craig Rogers et al. «Complications of Laparoscopic and Robotic Nephron Sparing Surgery». In: *Journal of the Society of Laparoendoscopic Surgeons* (Jan. 2011), p. 1.

[34] Ugur Boylu et al. «Comparison of surgical, functional, and oncological outcomes of open and robot-assisted partial nephrectomy». In: *Journal of minimal access surgery*. 2015.

[35] Brian M. Benway et al. «Robot Assisted Partial Nephrectomy Versus Laparoscopic Partial Nephrectomy for Renal Tumors: A Multi-Institutional Analysis of Perioperative Outcomes». In: *The Journal of Urology* 182.3 (2009), pp. 866–873. ISSN: 0022-5347.

[36] Steven M. Lucas et al. «A Comparison of Robotic, Laparoscopic and Open Partial Nephrectomy». In: *JSLS : Journal of the Society of Laparoendoscopic Surgeons*. 2012.

[37] Lee CU, Kang M, and Sung HH et al. «Comparison of 5-Year Outcomes of Robot-Assisted Laparoscopic and Laparoscopic Partial Nephrectomy in Patients With Localized Renal Cell Carcinoma». In: *Korean journal of urological oncology* 15.3 (2017), pp. 172–177.

[38] Veronica Penza et al. «Long Term Safety Area Tracking (LT-SAT) with Online Failure Detection and Recovery for Robotic Minimally Invasive Surgery». In: *Medical Image Analysis* 45 (Dec. 2017).

[39] Stuart A. Bowyer, Brian L. Davies, and Ferdin Rodriguez y Baena. «Active Constraints/Virtual Fixtures: A Survey». In: *IEEE Transactions on Robotics* 30 (Feb. 2014), pp. 138–157.

[40] L.B. Rosenberg. «Virtual fixtures: Perceptual tools for telerobotic manipulation». In: *IEEE Annual Virtual Reality International Symposium*. Oct. 1993, pp. 76–82. ISBN: 0-7803-1363-1.

[41] Shahram Payandeh and Zoran Stanisic. «On application of virtual fixtures as an aid for telemanipulation and training». In: *Haptic Interfaces for Virtual Environment and Teleoperator Systems*. Feb. 2002, pp. 18–23. ISBN: 0-7695-1489-8.

[42] Ming Li, Ankur Kapoor, and R.H. Taylor. «A constrained optimization approach to virtual fixtures». In: *International Conference on Intelligent Robots and Systems*. Sept. 2005, pp. 1408–1413.

[43] Catherina Burghart et al. «Robot Controlled Osteotomy in Craniofacial Surgery». In: *Workshop on Haptic Devices in Medical Applications* (June 2004).

[44] Alessandro Bettini et al. «Vision-Assisted Control for Manipulation Using Virtual Fixtures». In: *IEEE Transactions on Robotics* 20 (Jan. 2005), pp. 953–966.

[45] Alex B. Kuang et al. «Assembling Virtual Fixtures for Guidance in Training Environments». In: *12th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS 2004), 27-28 March 2004, Chicago, IL, USA, Proceedings*. 2004, pp. 367–374.

[46] Ryo Kikuuwe et al. «Admittance and Impedance Representations of Friction Based on Implicit Euler Integration». In: *IEEE Transactions on Robotics* 22 (Dec. 2006), pp. 1176–1188.

[47] N. Enayati et al. «Robotic Assistance-as-Needed for Enhanced Visuomotor Learning in Surgical Robotics Training: An Experimental Study». In: *IEEE International Conference on Robotics and Automation* (2018).

[48] Morgan Quigley et al. «ROS: an open-source Robot Operating System». In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) Workshop on Open Source Robotics*. 2009.

[49] *Qt: Cross-platform software development for embedded & desktop*. URL: `https://www1.qt.io/`.

[50] *Bullet Real-Time Physics Simulation*. URL: `https://pybullet.org/`.

[51] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit (4th ed.)* Kitware, ISBN: 978-1-930934-19-1, 2006.

[52] Radu Bogdan Rusu and Steve Cousins. «3D is here: Point Cloud Library (PCL)». In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2011.

[53] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, 2004.

[54] R. Courant and H. Robbins. *What Is Mathematics? An Elementary Approach to Ideas and Methods*. Oxford University Press, ISBN: 9780195005172, 1996.

[55] W. Randolph Franklin. *PNPOLY - Point Inclusion in Polygon Test.* 1970. URL: `https://wrf.ecse.rpi.edu//Research/Short_Notes/pnpoly.html`.

[56] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. «Poisson Surface Reconstruction». In: *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*. SGP '06. Eurographics Association, 2006, pp. 61–70. ISBN: 3-905673-36-3.

[57] Bertram H. Drost and Slobodan Ilic. «Almost constant-time 3D nearest-neighbor lookup using implicit octrees». In: *Machine Vision and Applications* 29.2 (Feb. 2018), pp. 299–311. ISSN: 1432-1769.

[58] William E. Lorensen and Harvey E. Cline. «Marching Cubes: A high resolution 3D surface construction algorithm». In: *Computer Graphics* 21.4 (1987), pp. 163–169.

Article written by the author during the thesis

[59] Luca Vantadori et al. «Intraoperative Safety Constraints Definition System for Robotic Minimally Invasive Surgery». In: *CRAS: Joint Workshop on New Technologies for Computer/Robot Assisted Surgery* (2017).

Pictures source

[1] Taken from: https://www.chronogram.com/hudsonvalley/recent-advances-in-laparoscopic-surgery.

[2] Taken from: http://www.hysterectomy.org/what-is-a-hysterectomy.

[3] Taken from: https://infodrones.it/vinci-robot-bisturi/.

[4] Taken from: https://www.intuitivesurgical.com.

[5] Taken from: https://www.cancer.org/cancer/prostate-cancer.

[6] Taken from: https://www.cancer.org/cancer/prostate-cancer/treating/surgery.html.

[7] Taken from: https://www.martinhealth.org/prostate-surgery.

[8] Taken from: https://www.indiamart.com/proddetail/veress-needle.html.

[9] Taken from: http://drfarrahcancercenter.com/kidney-cancer/.

[10] Adapted from: https://www.martinhealth.org/partial-nephrectomy-robotic-surgery.

[11] Taken from: https://www.memorialmedical.com/services/da-vinci-robotic-surgery/urology.

[12] Taken from: https://www.semanticscholar.org/paper/Identification-of-inferior-pancreaticoduodenal.

[13] Adapted from: https://earlsview.com.

[14] Adapted from: https://www.semanticscholar.org/paper/Active-Constraints-Virtual-Fixtures-A-Survey-Bowyer-Davies.

[15] Taken from: http://robohub.org.

[16] Taken from: https://github.com/neemoh/ATAR.

[17] Taken from: https://docs.opencv.org.

[18] Adapted from: https://www.thedoctorsclinic.com.

[19] Adapted from: https://medium.com/retronator-magazine.

[20] Adapted from: https://www.cse.unr.edu.

[21] Taken from: https://doi.org/10.1007/s00138-017-0889-4.

[22] Taken from: http://www.smartsurg-project.eu.

[23] Taken from: http://hhoppe.com/poissonrecon.pdf.