SARAS CALL h2020-ICT-2016-2017

INFORMATION AND COMMUNICATION TECHNOLOGIES

SARAS

"Smart Autonomous Robotic Assistant Surgeon"

D3.1 – Multi-modal human-robot interfaces and architecture for the MULTIROBOTS-SURGERY platform

Due date of deliverable: December 31, 2018

Actual submission date: December 31, 2018

Grant agreement number: 779813
Start date of project: 01/01/2018

Lead contractor: Università di Verona
Duration: 36 months

Project funded by the European Commission within the EU Framework Programme for Research and Innovation HORIZON 2020	
Dissemination Level	
PU = Public, fully open, e.g. web	✓
CO = Confidential, restricted under conditions set out in Model Grant Agreement	
CI = Classified, information as referred to in Commission Decision 2001/844/EC	
Int = Internal Working Document	

D3.1 – Multi-modal human-robot interfaces and architecture for the MULTIROBOTS-SURGERY platform

Editors

Federica Ferraguti (UNIMORE)
Marco Minelli (UNIMORE)
Cristian Secchi (UNIMORE)

Contributors

Alícia Casals (UPC)
Albert Hernansanz (UPC)
Narcís Sayols (UPC)
Alessio Sozzi (UNIFE)
Marcello Bonfè (UNIFE)
Giacomo De Rossi (UNIVR)
Francesco Setti (UNIVR)
Riccardo Muradore (UNIVR)

Reviewers

All partners

The work described in this document has been conducted within the project SARAS, started in January 2018. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No.779813.

TABLE OF CONTENTS

List of Figures

Executive Summary

This document is intended to describe the multi-modal human-robot interface for the assistant surgeon and the control architecture for the SARAS assistive robotic arms in the multi-master/multi-slave teleoperated system, including workload experiments with the MULTIROBOTS-SURGERY platform.

1 Introduction

This deliverable is intended to describe the Human-Robot interface and teleoperation architecture for the assistant surgeon and the multi-master/multi-slave bilateral teleoperation architecture being implemented in the SARAS project.

2 Human-Robot interface and teleoperation architecture for the assistant surgeon

In this section, the control and perceptual architecture for the teleoperation of the assistive robotic arms will be described. The multimodal feedback for the assistant surgeon has been implemented using commercial haptics devices (Geomagic Touch) attached at two Simball joysticks and a head mounted display (Oculus Rift). The force experienced by the off-the-shelf laparoscopic tools mounted on the SARAS assistive robotic arms has been mapped to be consistent with the kinematic structure of the corresponding master side. The control architecture has been designed and validated to be robust and safe also in case of communication delays between masters and slaves.

2.1 Bilateral teleoperation architecture for the assistant surgeon

The goal of SARAS is to develop an autonomous robotic assistant for a surgeon doing laparoscopic surgery. For achieving this objective, it is necessary to capture the behavior of the human assistant for reproducing and generalizing it. To this aim, we developed a multi-arms bilateral teleoperation system for allowing the assistant surgeon to tele-assist the main surgeon. This section describes the Multi-Master-Multi-Slave (MMMS) bilateral teleoperation architecture for allowing tele-assisted laparoscopic surgery. The system had to guarantee a stable interaction with a poorly known environment (i.e. the human body) and it needed to be flexible, in order to change the kind of feedback that can be provided to the user. In fact, since SARAS involves the robotic surgical scenario, information are continuously collected (e.g. by the endoscope) and processed. Thus, new information about the environment can be available during the operation and the force feedback provided to the user had to be updated accordingly. The bilateral teleoperation architecture that has been developed is an extension of the two layer architecture proposed in [10] to the MMMS teleoperation.

2.1.1 Master and slave sides

We consider a system composed by $N_m = 2$ masters (two Geomagic Touch devices attached to two Simball joysticks) and $N_s = 2$ slave robots (MEDINEERING robots). Each robot can be modeled as the following Euler-Lagrange system:

$$\Lambda_{w_i}\left(x_{w_i}(t)\right)\ddot{x}_{w_i}(t) + \mu_{w_i}\left(x_{w_i}(t), \dot{x}_{w_i}(t)\right)\dot{x}_{w_i}(t) = F^{\tau}_{w_i}(t) + F^{ext}_{w_i}(t) \tag{1}$$

where $w \in \{m, s\}$ where the subscripts m and s indicate the master and the slave side, respectively, and $i = 1, \ldots, N_w$, $x_{w_i}(t)$ are the coordinates of the configuration of the end-effector in the task space, $\lambda_{w_i}(x_{w_i}(t))$ is the symmetric and positive-definite inertia matrix and $\mu_{w_i}(x_{w_i}(t), \dot{x}_{w_i})$ is the Coriolis/centrifugal matrix. The term $F^{\tau}_{w_i}(t)$ represents the control inputs while $F^{ext}_{w_i}(t)$ is the vector of generalized external forces, i.e. the force applied by the user or the force applied by the environment. It is possible to build a Euler-Lagrangian model of the whole master and slave sides.

Defining

$$\begin{aligned} x_w(t) &= \left[x_{w_1}^T(t), \ldots, x_{w_{N_w}}^T(t)\right]^T, \\ \Lambda_w(x_w(t)) &= diag\left[\Lambda_{w_1}, \ldots, \Lambda_{w_{N_w}}\right], \\ \mu_w(x_w(t), \dot{x}_w(t)) &= diag\left[\mu_{w_1}, \ldots, \mu_{w_{N_w}}\right], \\ F_w^\tau(t) &= \left[F_{\tau^T}(t), \ldots, F_{w_{N_w}}^{\tau^T}(t)\right]^T, \\ F_w^{ext}(t) &= \left[F_{ext^T}(t), \ldots, F_{w_{N_w}}^{ext^T}(t)\right]^T, \end{aligned} \tag{2}$$

and exploiting (1) we can model each side of the teleoperation system as the following Euler-Lagrange system.

$$\Lambda_w(x_w(t))\ddot{x}_w(t) + \mu_w(x_w(t), \dot{x}_w(t))\dot{x}_w(t) = F_w^\tau(t) + F_w^{ext}(t) \tag{3}$$

The kinetic energy of the system described in (3) is given by the sum of the kinetic energies of all the robots in the w side and, compactly, it is defined as

$$V_w\left(\dot{x}_w(t)\right) = \frac{1}{2}\dot{x}_w(t)^T\Lambda_w\left(x_w(t)\right)\dot{x}_w(t) \tag{4}$$

The passivity of the multi-robot system, either at the master or at the slave sides, can be easily shown. In fact, from (4) we can write

$$\dot{V}_w(t) = \dot{x}_w(t)^T\Lambda_w(t)\ddot{x}_w(t) + \frac{1}{2}\dot{x}_w(t)^T\dot{\Lambda}_w(t)\dot{x}_w(t) \tag{5}$$

Computing $\ddot{x}_w(t)$ from (3), replacing it in (5) and considering that $\dot{\lambda}_w(t) - 2\mu_w(t)$ is skew symmetric one obtains:

$$\dot{V}_w(t) = \dot{x}_w^T(t)\left(F_w^\tau(t) + F_w^{ext}(t)\right) \tag{6}$$

which implies that

$$\int_0^t \dot{x}_w^T(\tau)\left(F_w^\tau(\tau) + F_w^{ext}(\tau)\right)d\tau \geq -V_w(0) \tag{7}$$

which means that the system is passive.

In the following will be described the MMMS teleoperation architecture that has been developed to allow to share the energy between the components of each side of the system in order to increase the flexibility of the overall architecture.

2.1.2 Shared energy tanks

In order to passively implement the bilateral teleoperation architecture, the model of each side of the system is augmented introducing an energy tank at both master and slave sides. Firstly introduced in [5] and then used in a wide range of applications like multi-robot systems [9] and surgical robotics [8], the energy tank is used to store the energy dissipated by a system and then to re-use this energy to implement the control action, without violating passivity.

We implemented on each robot a controlled dissipation in order to be able to harvest some energy for filling the energy tank when necessary. Then, a shared energy tank is connected to all the robots on the same side. Formally, we split the control input of each robot into the sum of two terms:

$$F_{w_i}^\tau = \phi_{w_i}^d + \phi_{w_i}^t \tag{8}$$

The first term is exploited for implementing a variable local damping by setting $\phi^d_{w_i} = -D_{w_i}(t)\dot{x}_{w_i}(t)$, where $D_{w_i}(t)$ is a time-varying positive semi-definite matrix. By embedding the damping injection into (1) we get the following damped Euler-Lagrange model for each robot.

$$\Lambda_{w_i}(x_{w_i})\ddot{x}_{w_i} + \mu_{w_i}(x_{w_i}, \dot{x}_{w_i})\dot{x}_{w_i} + D_{w_i}\dot{x}_{w_i} = \phi^t_{w_i} + F^{ext}_{w_i} \tag{9}$$

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

$$\begin{cases} \dot{x}_{t_w} = \frac{\sigma_w}{x_{t_w}} \sum_{i=1}^{N_w} \dot{x}^T_{w_i} D_{w_i}(t)\dot{x}_{w_i} + u_{t_w} \\ y_{t_w} = \frac{\partial T_w}{\partial x_{t_w}} \end{cases} \tag{10}$$

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

$$T_w(x_{t_w}) = \frac{1}{2}x^2_{t_w} \tag{11}$$

is the energy stored in the tank. Using 11 with 10 it is easy to see that:

$$\dot{T}_w(x_{t_w}) = \sigma_w \sum_{i=1}^{N_w} \dot{x}^T_{w_i} D_{w_i}(t)\dot{x}_{w_i} + u_{t_w} y_{t_w} \tag{12}$$

namely, that the tank stores all the energy dissipated by the robots using local damping injection and that energy can be injected/extracted from the port (u_{t_w}, y_{t_w}). Each robot is interconnected to the energy tank in order to exploit the energy stored in the tank for implementing the desired input. This can be done by the following power preserving interconnection between all the robots and the shared energy tank

$$\begin{cases} \phi^t_{w_i} = \omega_{w_i} y_{t_w} \\ u_{t_w} = -\sum_{i=1}^{N_w} \omega^T_{w_i} \dot{x}_{w_i} \end{cases} \qquad i = 1, \dots, N_w \tag{13}$$

Since

$$\sum_{i=1}^{N_w} \dot{x}^T_{w_i} \phi^t_{w_i} = -u_{t_w} y_{t_w} \tag{14}$$

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

By grouping (9) and by considering (10) and (13), it is possible to model each side of the MMMS teleoperation system as:

$$\begin{cases} \Lambda_w \ddot{x}_w + \mu_w \dot{x}_w + D_w \dot{x}_w = \omega_w x_{t_w} + F^{ext}_w \\ \dot{x}_{t_w} = \dfrac{\sigma_w}{x_{t_w}} \dot{x}^T_w D_w \dot{x}_w - \omega^T_w \dot{x}_w \end{cases} \tag{15}$$

where

$$D_w(t) = diag\{D_{w_1}(t), \dots, D_{w_{N_w}}(t)\} \tag{16}$$

and

$$\omega_w = [\omega_{w_1}, ..., \omega_{w_{N_w}}]^T \tag{17}$$

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

$$\sigma_w = \begin{cases} 1 & if\ T(x_{t_w}) \leq T_w^{max} \\ 0 & otherwise \end{cases} \tag{18}$$

where T_w^{max} represents the energy upper bound. Indeed, it is necessary to avoid excessive energy storing in the tank that may allow the implementation of practically unstable behaviors [12]. If $x_{t_w(t)} = 0$, i.e. when no more energy is left in the tank, the (15) becomes singular. To avoid this problem we initialize the tank with some energy and we prevent energy extraction below a certain treshold of the tank. Thus, we choose $x_{t_w}(0)$ such that $T_w(x_{t_w}(0)) \geq T_w^{min}$, where $T_w^{min} > 0$ is the minimum amount of energy that needs to be stored in the tank.

If there is enough energy in the tank the local damping injections of the robots are set to a minimum level $D_{w_i}^{min}$ while when the level of the tank is going below a certatain threshold $T_w^{min} < T_w^R < T_w^{max}$, the damping injection terms are increased to harvest energy. Thus we have that $D_{w_i}(t) = \xi(T_w(t))$, where $\xi(T) : \mathbb{R} \to \mathbb{R}$ is any smooth non decreasing function such that $D_{w_i}(t) = D_w^{min}$ if $T_w(t) > T_w^R$ and $D_{w_i}(t) = D_w^{max}$ if $T_w(t) \leq T_w^R$. Thus, we have that

$$D_w(t) = \begin{cases} D_w^{min} & T(x_{x_w}) > T_w^R \\ D_w^{max} & T(x_{x_w}) \leq T_w^R \end{cases} \tag{19}$$

where $D_w^{min} = diag(D_{w_1}^{min}, \dots D_{w_{N_w}}^{min})$ and $D_w^{max} = diag(D_{w_1}^{max}, \dots D_{w_{N_w}}^{max})$. The constants T_w^{min}, T_w^{max}, T_w^R, $D_{w_i}^{min}$, $D_{w_i}^{max}$ are application dependent design parameters. As evident from (19), all the robots contribute in the same way to the energy harvesting. Nevertheless, robot specific damping strategies may be designed.

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

$$\omega_{w_i} = \begin{cases} \frac{F_{w_i}^d}{x_{t_w}} & if\ T(x_{t_w}) \geq T_w^R \\ K_w(T(x_w))\frac{F_{w_i}^d}{x_{t_w}} & \text{otherwise} \end{cases} \tag{20}$$

where

$$K_w(T(x_w)) = \max\left(0, \frac{T(x_w) - T_w^{min}}{T_w^R - T_w^{min}}\right) \tag{21}$$

Thus, if there is enough energy in the tank, the desired input $F_{w_i}^d$ is implemented otherwise only a scaled version of the desired input is implemented. In the worst case $K_w(T(x_w)) = 0$ and, therefore, nothing will be implemented in order to preserve passivity. Nevertheless, since the local damping is set to its maximum when $T(x_{t_w}) < T_w^R$, its very unlikely that $K_w(T(x_w)) = 0$ in practice. Figure 1 shows the coupling of two generic master or slave devices with the energy tank. The augmented model in (15) consists of the system (3) with the damping element $D_w(x_{t_w})$ and the tank that is energetically coupled through the input ω_{w_i}. The kinetic energy is still given by (4).

However, using the same procedure that lead from (4) to (6), for the augmented model it follows that

$$\dot{V}_w(t) = \dot{x}_w^T \omega_w x_{t_w} + \dot{x}_w^T F_w^{ext} - \dot{x}_w^T D_w \dot{x}_w \tag{22}$$

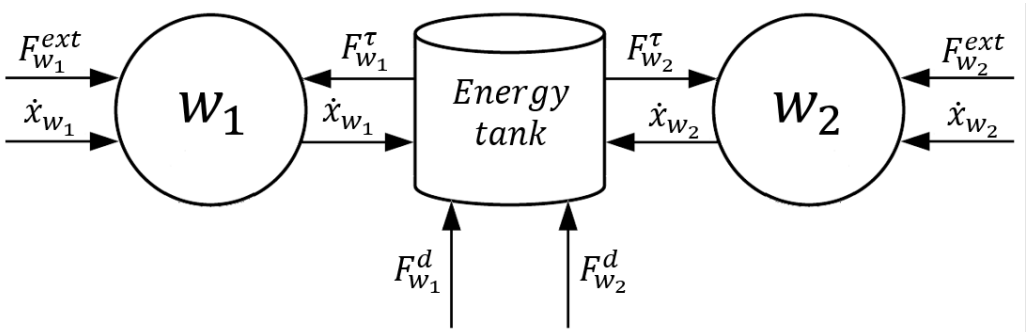

Figure 1: Coupling of two generic master or slave devices w_1 and w_2 with the energy tank used to store the dissipated energy.

Furthermore, by plugging (13) in (12) and by considering the definition of D_w in (15), we get

$$\dot{T}_w(t) = \sigma_w(\dot{x}_w^T D_w \dot{x}_w) - x_{t_w}\omega_w^T \dot{x}_w \tag{23}$$

It can be proved that the system in (15) is passive with respect to the pair $((F^{ext}_{w_1},...,F^{ext}_{w_{N_w}}), (\dot{x}_{w_1},..., \dot{x}_{w_{N_w}}))$. Indeed, consider as a storage function the total energy of the teleoperation system 15:

$$\mathcal{V}(t) = V_w(t) + T_w(t) \tag{24}$$

where $V_w(t)$ represents the energy associated to the master or slave device under consideration and T_w the energy stored in the tank. From (24) it follows that

$$\dot{\mathcal{V}}(t) = \dot{V}_w(t) + \dot{T}_w(t) \tag{25}$$

Substituting (22) and (23) in (25) we obtain

$$\dot{\mathcal{V}}(t) = \dot{x}_w^T F_w^{ext} - (1-\sigma_w)\dot{x}_w^T D_w \dot{x}_w \tag{26}$$

Since $\sigma_w \in \{0, 1\}$, we have that

$$\dot{\mathcal{V}}(t) \leq \sum_{i=1}^{N_w} \dot{x}_{w_i}^T F_{w_i}^{ext} \tag{27}$$

which implies the following passivity condition:

$$\mathcal{V}(t) - \mathcal{V}(0) \leq \int_0^t \sum_{i=1}^{N_w} \dot{x}_{w_i}^T(\tau) F_{w_i}^{ext}(\tau) d\tau \tag{28}$$

2.1.3 The bilateral control architecture

In this section the proposed control architecture for the bilateral teleoperation of a multi-master-multi-slave system is considered. In order to interconnect master and slave sides of a delayed communication channel, we endowed each tank with two power inputs, as proposed in [8]. The overall architecture is shown in Figure 2 and it can be decomposed into two layers: a *Transparency Layer* and a *Passivity Layer*. In the figure, it is shown the SARAS architecture, where two master devices ($N_m = 2$) and two slave devices ($N_s = 2$) are used.

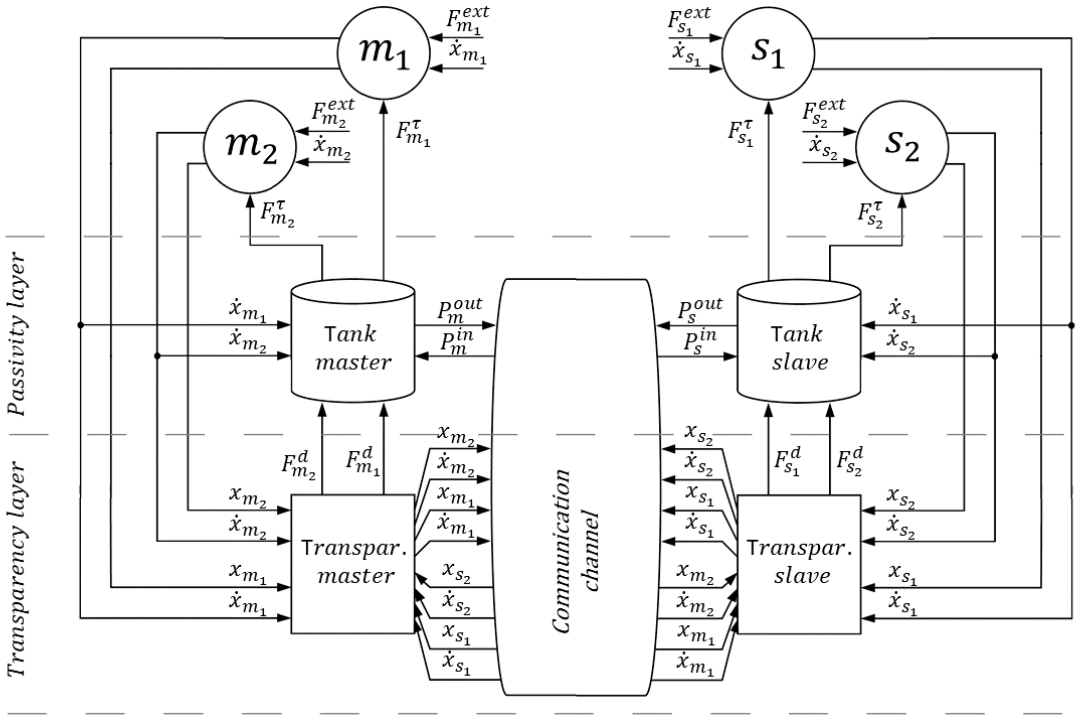

Figure 2: Coupling of two generic master devices m_1 and m_2 with two slave devices s_1 and s_2 and one tank per side by means of the communication channel

First of all, each side of the teleoperation system is augmented with a tank (*Tank master* and *Tank slave* in Fig. 2), as described in Section 2.1.1. In the Transparency Layer, master and slave devices exchange position, velocity and force information that are used for computing the desired inputs ($F^d_{m_1}$, $F^d_{m_2}$, $F^d_{s_1}$, $F^d_{s_2}$), as it will be shown later. In order to keep Fig. 2) simple, the forces exchange is not shown in the picture. These forces are sent to the Passivity Layer, whose role is to passively implement them using the energy stored in the tanks, as it will be formally proved. Master and slave energy tanks can exchange power for balancing the amount of energy stored at master and slave sides. Formally, the overall architecture with N_m master devices, N_s slave devices and one tank per side can be modeled as

$$\begin{cases} \Lambda_m \ddot{x}_m + \mu_m \dot{x}_m + D_m \dot{x}_m = \omega_m x_{t_m} + F^{ext}_m \\ \dot{x}_{t_m} = \dfrac{\sigma_m}{x_{t_m}} \dot{x}^T_m D_m \dot{x}_m + \dfrac{1}{x_{t_m}} (\sigma_m P^{in}_m - P^{out}_m) - \omega^T_m \dot{x}_m \\ \Lambda_s \ddot{x}_s + \mu_s \dot{x}_s + D_s \dot{x}_s = \omega_s x_{t_s} + F^{ext}_s \\ \dot{x}_{t_s} = \dfrac{\sigma_s}{x_{t_s}} \dot{x}^T_s D_s \dot{x}_s + \dfrac{1}{x_{t_s}} (\sigma_s P^{in}_s - P^{out}_s) - \omega^T_s \dot{x}_s \end{cases} \tag{29}$$

where $P^{in}_m, P^{in}_s \geq 0$ and $P^{out}_m, P^{out}_s \geq 0$ are incoming and outgoing power flows that the tanks can exchange with each other by means of the communication channel. The policy for defining P^{out}_m, P^{out}_s is the same as the one reported in [8]. In presence of time delay Δt between the two sides we have that

$$\begin{cases} P^{in}_s(t) = P^{out}_m(t - \Delta t) \\ P^{in}_m(t) = P^{out}_s(t - \Delta t) \end{cases} \tag{30}$$

While the power is traveling from one side to the other, it is stored in the communication channel

that becomes an energy storing element in the teleoperation system. In particular, as shown in [8], we have that

$$H_{ch}(t) = \int_{t-\Delta t}^{t} P_m^{out}(\tau) + P_s^{out}(\tau) d\tau \tag{31}$$

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

To interact with the system and then with the environment, the transparency layer provides each side with the desired set-points. In the surgical scenario considered in this paper, the transparency layer can be modeled as follow:

$$\begin{cases} F_{m_j}^d = F_{ca_j} + F_{s_j}^{ext} \\ F_{s_z}^d = -K_p(x_{m_z} - x_{s_z}) - K_d(\dot{x}_{m_z} - \dot{x}_{s_z}) \end{cases} \tag{32}$$

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

The collision avoidance forces $F_{ca_j}(t)$ have been introduced as virtual fixtures [7] to avoid collisions between the arms of the slave robots that share the same workspace. In the SARAS scenario the pivoting movement of the laparoscopic tools need to be always guaranteed. For this reason, the collision avoidance is achieved providing force feedback only at the master side, so that the motion of each slave arm still remains constrained to the pivoting point by the mechanics of the master device. Collisions are avoided following three different steps:

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

It can be proven that the strategy illustrated so far guarantees the passivity of the teleoperation system (29) with respect to the pair $((F_{m_1}^{ext},...,F_{m_{N_m}}^{ext}, F_{s_1}^{ext},...,F_{s_{N_s}}^{ext}),(\dot{x}_{m_1},...,\dot{x}_{m_{N_m}},\dot{x}_{s_1},...,\dot{x}_{s_{N_s}}))$.

Indeed, consider as a storage function the total energy of the teleoperation system:

$$W(t) = V_m(t) + V_s(t) + T_m(t) + T_s(t) + H_{ch}(t) \tag{33}$$

Using (29) we have that

$$\begin{aligned} \dot{W}(t) = \dot{x}_m^T F_m^{ext} - \dot{x}_m^T D_m \dot{x}_m + \dot{x}_s^T F_s^{ext} - \dot{x}_s^T D_s \dot{x}_s + \\ + \sigma_m(\dot{x}_m^T D_m \dot{x}_m) + \sigma_m P_m^{in}(t) - P_m^{out}(t) + \\ + \sigma_s(\dot{x}_s^T D_s \dot{x}_s) + \sigma_s P_s^{in}(t) - P_s^{out}(t) + \dot{H}_{ch} \end{aligned} \tag{34}$$

From (31) we have that

$$\dot{H}_{ch}(t) = P_m^{out}(t) - P_m^{out}(t - \Delta t) + P_s^{out}(t) - P_s^{out}(t - \Delta t) \tag{35}$$

Considering (30) and replacing (31) in (34) we have that:

$$\begin{aligned}\dot{W}(t) = \dot{x}_m^T F_m^{ext} + \dot{x}_s^T F_s^{ext} - (1-\sigma_m)\dot{x}_m^T D_m \dot{x}_m + \\ - (1-\sigma_s)\dot{x}_s^T D_s \dot{x}_s - (1-\sigma_m)P_s^{out}(t-\Delta t) + \\ - (1-\sigma_s)P_m^{out}(t-\Delta t)\end{aligned} \tag{36}$$

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

$$\dot{W}(t) \leq \dot{x}_m^T F_m^{ext} + \dot{x}_s^T F_s^{ext} \tag{37}$$

whence

$$\dot{W}(t) \leq \sum_{j=1}^{N_m} \dot{x}_{m_j}^T F_{m_j}^{ext} + \sum_{z=1}^{N_s} \dot{x}_{s_z}^T F_{s_z}^{ext} \tag{38}$$

which implies the following passivity condition:

$$W(t) - W(0) \leq \int_0^t \left(\sum_{j=1}^{N_m} \dot{x}_{m_j}^T(\tau) F_{m_j}^{ext}(\tau) + \sum_{z=1}^{N_s} \dot{x}_{s_z}^T(\tau) F_{s_z}^{ext}(\tau) \right) d\tau \tag{39}$$

2.1.4 Preliminary validation

A preliminary validation of the bilateral teleoperation for the assistant surgeon has been performed at UNIMORE. The experimental results are reported in the following and they are organized into two different experiments: the first experiment shows the main performances of the teleoperation system while the second experiment highlights the main disadvantages of the standard two-layer approach. All the experiments are performed introducing a time delay $\Delta t = 300$ ms between the master and the slave side in order to show the robustness of the control architecture

In order to faithfully replicate the SARAS scenario and to provide force feedback to the surgeon, a pair of ad-hoc teleoperation devices composed by a non-actuated 4 DOF Simball joystick laparoscopic haptic device coupled with a 6 DOF Geomagic Touch haptic device are placed at the master side. At the slave side a KUKA LWR 4+ 7-DOF robot and a Universal Robots UR5 6-DOF manipulator endowed with 3D printed laparoscopic tools are used to physically interact with the environment. The setup is shown in Fig. 3

In the first part of the first experiment the laparoscopic tools on the slave robots were teleoperated in order to interact with a soft material, replicating a simplified interaction with human tissue. Then, the slave arms were moved trying to cause collisions, showing the action of the collision avoidance algorithm. For the sake of clarity and since the DOFs are usually chosen to be decoupled, we will show the plots of one translational DOF. Fig. 4 shows the Cartesian position of the master and of the slave devices.

The communication delay introduced in the control architecture causes an evolution of the tracking error characterized by a maximum value of $0.032m$ for the left side and of $0.014m$ for the right side. However, the average value turns out to be $0.0092m$ for the left side and $0.0046m$ for the right side. Fig. 5-a shows the forces exchanged on the master-left side that highlights a good behavior of the overall system. During the experiments the system was controlled in order to interact with

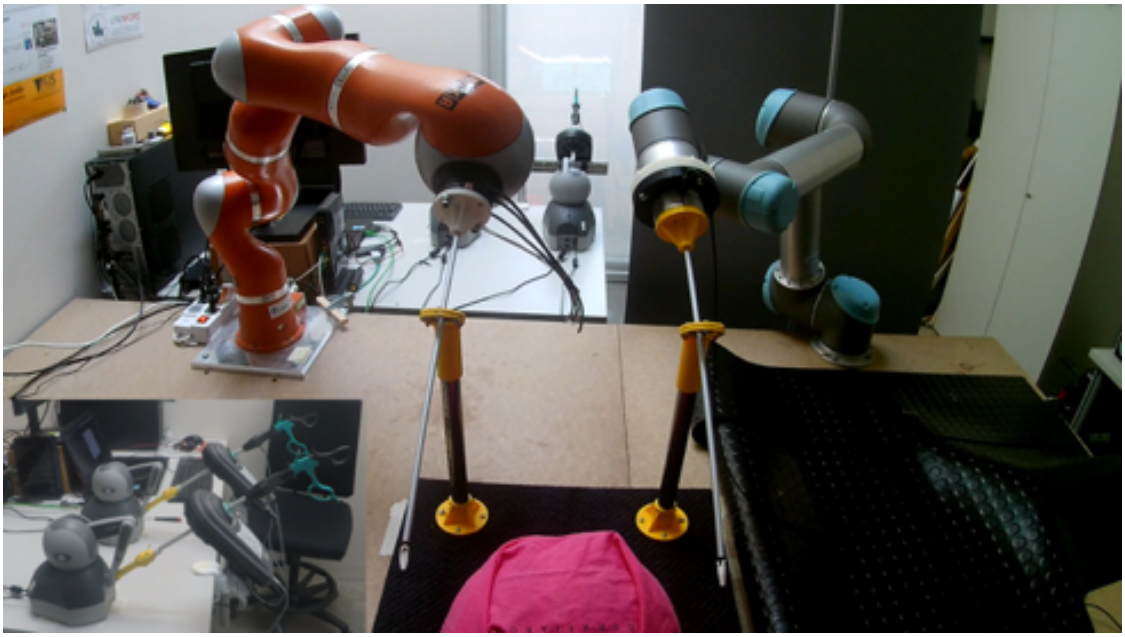

Figure 3: Setup for the preliminary validation of the bilateral teleoperation architecture for the assistant surgeon.

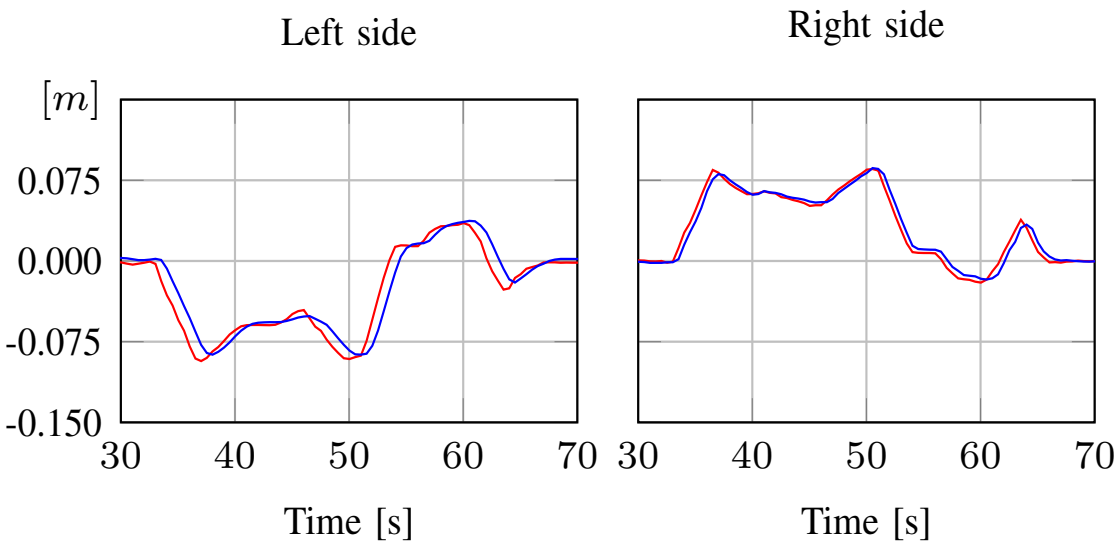

Figure 4: Cartesian position of the master devices (red line) and of the slave device (blue line) for the right side and the left side.

a physical object and Fig. 5-b shows the interaction forces measured by the force/torque sensor. Fig. 5-c reports the effect of the collision avoidance. In the ending part of the experiment can be observed how the repulsive force tries to guide the user to move the robots away after a critical configuration has been detected.

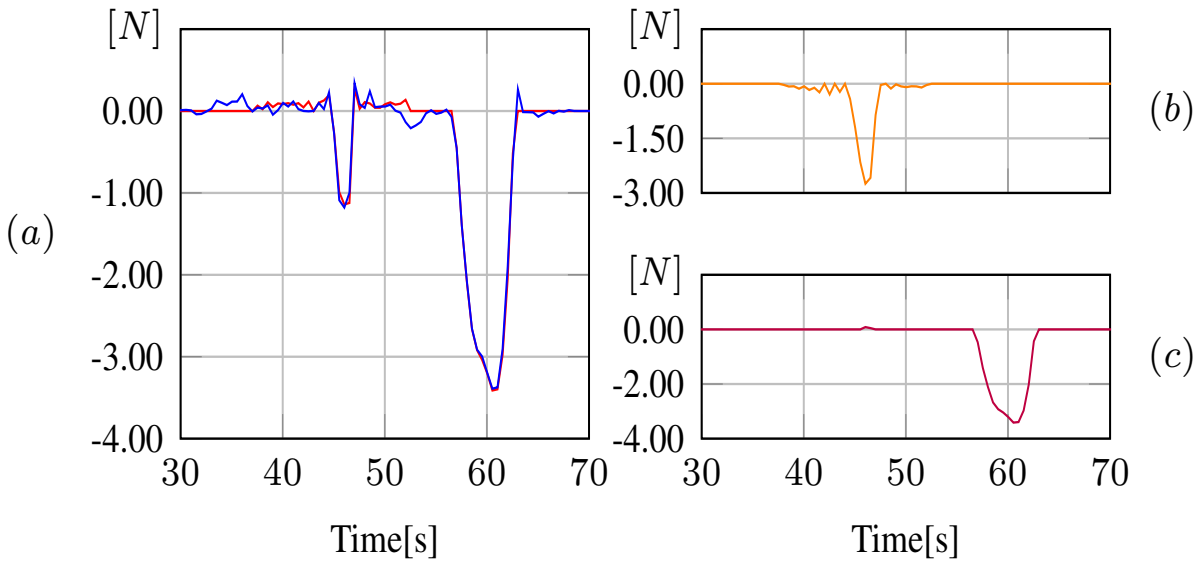

Figure 5: (a) Desired force (red line) and applied force (blue line) on the master devices at the left side. The desired force is given by the sum of (b) the measured force at the slave side (orange line) and (c) the collision avoidance action (purple line), properly translated to fit with the master device. The applied force depends on the available energy, reported in Figure 6, and is given by the desired force and the damping counterpart introduced by the tank

Figure 6 shows the dissipated energy stored in the master and slave tanks.

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

In the second experiment the teleoperation architecture was modified splitting the single master tank and the single slave tank into two independent master and slave tanks. The value of the initial energy and the damping coefficient introduced by the tank were also reduced. During the experiment, an external force was applied to the right slave arm causing the emptying of the tank and stopping the robot, as can be seen in Fig 7, where the evolution of the energy stored in the tanks and the cartesian position of the master and slave devices are shown. At time $47s$ the energy stored in the right tank goes to its lower bound, inhibiting the movement of the robot, as highlighted by the shadowed red area in Fig. 7 . Looking at the cartesian position of the robot compared to the one commanded by the master side it is possible to notice that the arm stills in the same position while the master device moves, increasing the tracking error. This behavior does not affect the left side which continues to work without issues. In such a case using the shared energy approach would

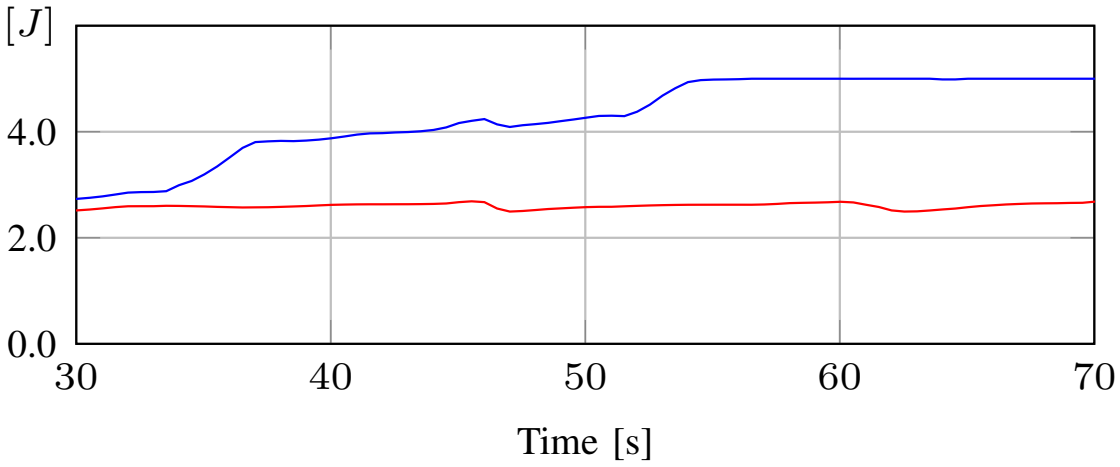

Figure 6: Dissipated energy stored in the energy tank at the master side (red line) and at the slave side (blue line).

have made the energy of the left robot available to the right robot, avoiding this kind of situation.

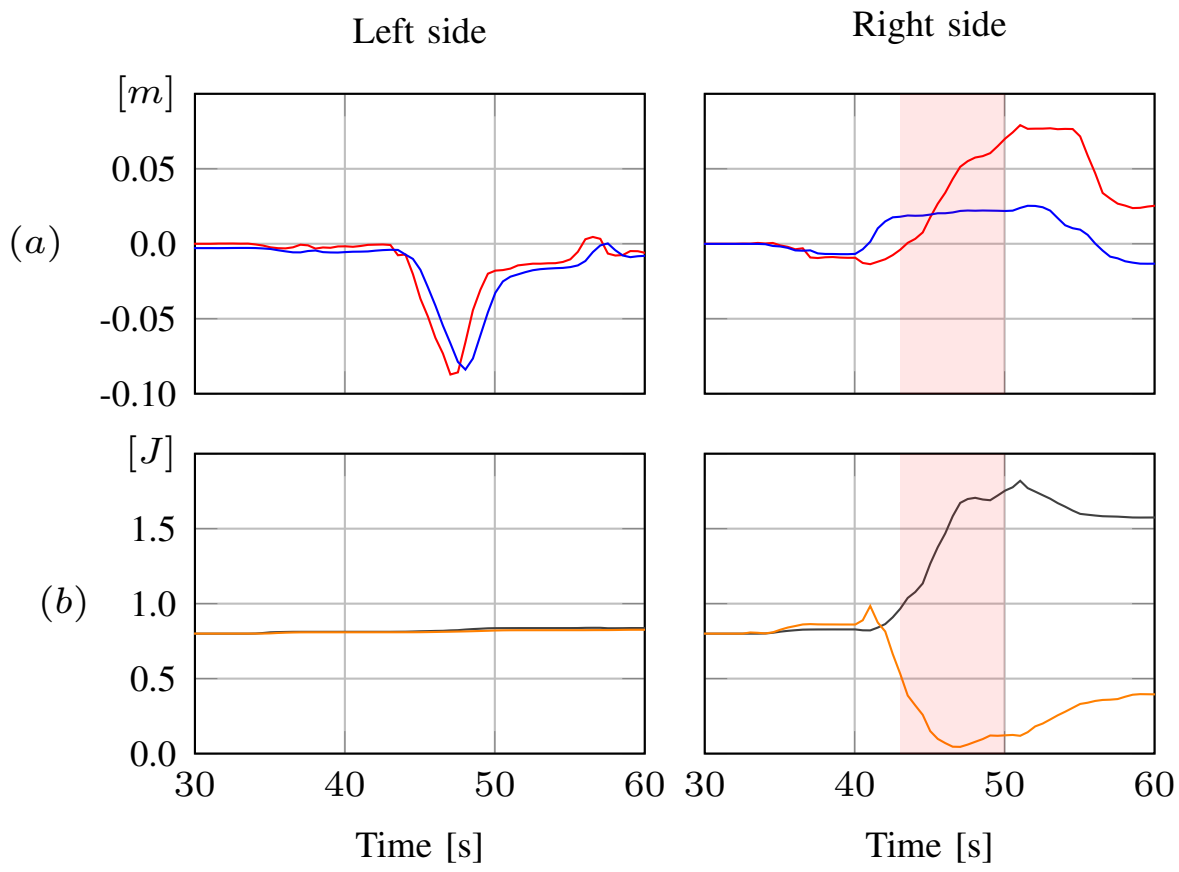

Figure 7: (a) Cartesian position of the master device (red line) and of the slave device (blue line) for the right side and the slave side. (b) Energy stored in the master tank (grey line) and in the slave tank (orange line) for the right side and the slave side. The shadowed area underlines the time into which the right robot stuck.

2.1.5 Vision based force estimation

Despite all the benefits of Minimally Invasive Surgery (MIS), current commercially available systems suffer from one major limitation which is the lack of force feedback [21, 4]. This feature is of huge importance since it increases surgeon-patient transparency [17] and allows more natural interaction with delicate tissues. Without force feedback information, surgeons have no means of knowing how much force is applied to the tissue, which could complicate the surgical task, increase its completion time and, what is worst, result in irreversible injuries [3, 16]. Furthermore, dealing with the absence of this primary sense of touch creates a higher mental workout for surgeons and might be a hazardous source of distraction [13]. For these reasons, much effort is addressed to provide force feedback in robotic surgery. However, up to date it is still considered an open problem [2]. In the search for solutions for the lack of force feedback, some researchers have focused their efforts towards developing force sensing devices (FSDs) [22, 19, 6]. These devices can be placed either inside or outside the patient's body. When placed outside, the devices are attached to the robot or its instruments and offer indirect sensing. With this option, the devices measure not only the instrument-tissue interaction forces but also irrelevant force data given by the external/internal surgical environment. Removal of these undesirable measurements is not possible due to hysteresis and because they greatly depend on ambiguous starting conditions [15]. Alternatively, FSDs can offer direct sensing if they are placed close or on the tip of the instrument inside the patient's body. However, the internal location of the sensor introduces numerous problems, including: biocompatibility and sterilization constrains; long-term stability; adaption to surgical tool; size and high cost [11, 20]. All these limitations put severe restrictions to the adoption of FSDs in real surgical environments. An alternative to the use of FSDs is to compute the interaction forces by the observable deformation of the tissue in what is called Vision-based force estimation.

Our contribution to the present project is described in Fig. 8, following the work described [1]. Since all RAMIS settings include a videoscopic view of the operation workspace, we can employ the available visual information of the tool-tissue interaction and relate it directly to the applied force. From the conservation principles of continuum mechanics, it is clear that the change in shape of an elastic object is directly proportional to the force applied. Following this principle, we propose an approach that is based on a variational framework that allows computing the observable deformation after a force is applied. Then, this information is used in a learning system that finds the nonlinear relationship between the given data and use it to estimate the applied force.

The force estimation approach in development is based on a three steps process. First, a *Surface Reconstruction* that generates a dense point cloud corresponding the tissue surface. In the second step, *Deformation Computation*, the spatial relationship between these points allows the generation of a lattice that dynamically adapts to the tissue deformation as the points on the point cloud shift due to the applied force. Finally, in the third step, *Force Estimation*, a mapping relates the shifts of the lattice points, that is, the deformation, to the applied force.

Surface Reconstruction In this part of the algorithm, we deal with the reconstruction of the workspace. Our approach is based on the REMODE algorithm [18]. This algorithm estimates the depth map from a single moving camera using a probabilistic approach by combining a Bayesian estimation and convex optimizations for image processing. Furthermore, the algorithm exhibits high accuracy, efficiency in memory usage and, as this algorithm is implemented in CUDA, it is suitable for real time applications.

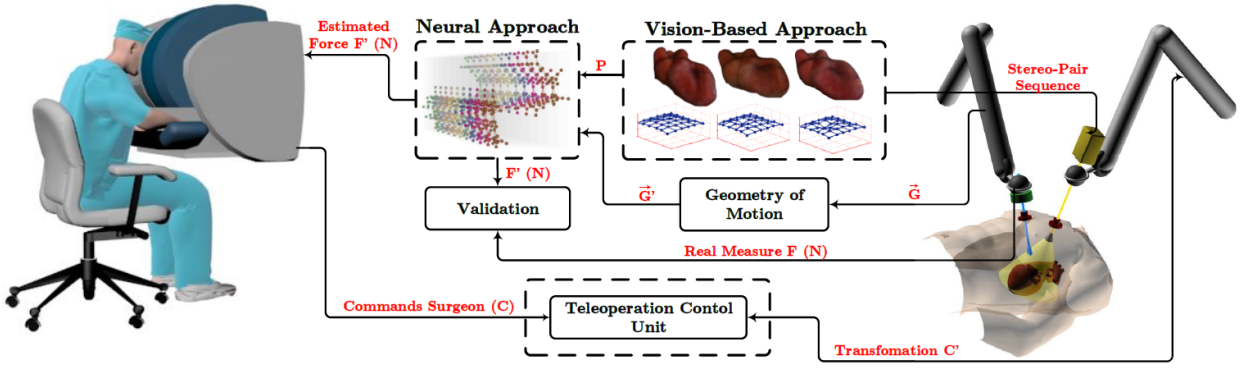

Figure 8: Flowchart of our approach for estimating applied forces in surgical robotic systems. We first propose a visual approach to reconstruct the surface, and then, minimizing an energy functional we compute the deformation structure over time. Then, deformation information, together with the geometry of motion, are used as input to an artificial neural network architecture which accurately estimates the applied force.

The work done in the reconstruction has been divided into the following parts:

- Adaptation of the work environment to the ROS system (100%): Another characteristic of the REMODE algorithm is that it is implemented as a ROS node. Therefore, we had to adequate our working environment to ROS by incorporating a KUKA LWR which has an interface compatible with ROS.
- Integration of the REMODE algorithm (90%): The incorporation of the REMODE algorithm to the force sensor system is practically finished.
- Test and validation of the algorithm (60%): A series of tests and validations of the algorithm has been carried out to test its performance. These tests suggest that the algorithm has suitable accuracy for the current problem.

Deformation Computation From the previous recovered surface in this step the algorithm computes the deformation in the surface as the changes produced in a set of linearly independent vectors (3D lattice) as can be seen in Fig. 9. We use this lattice as our deformation model in order to obtain a compromise between computation cost and accuracy, working with a reduced number of points. The 3D lattice Γ can be parametrized by the following formula:

$$\Gamma(x;P) = \sum_{l=1}^{y_1}\sum_{m=1}^{y_2}\sum_{n=1}^{y_3} P_{lmn} \prod_{k=1}^{K} \xi_k(x_k) \quad for\ k = 1, ..., 3 \tag{40}$$

where $\xi_k(x_k)$ are a cubic basis spline and P_{lmn} denotes the displacement of a control point with$y_1 y_2 y_3$ number of points. So, the lattice is calculated from the reconstruction by minimizing the following equation:

$$E_t(P) = E_\Phi(\Gamma(x;P), R) + \gamma E_\Psi(\Gamma(x;P)) + E_\Lambda(\Gamma(x;P)) \tag{41}$$

where E_Φ is the discrepancy measure term, γ is the regulation parameter, E_Ψ is the penalization term and E_Λ is a term to preserve the lattice shape.

Finally, the parameters P_{lmn} of the lattice are the inputs to the neural network used to determine the relation function between the deformation and the applied force of the last part of our approach. The work done in the deformation calculation are:

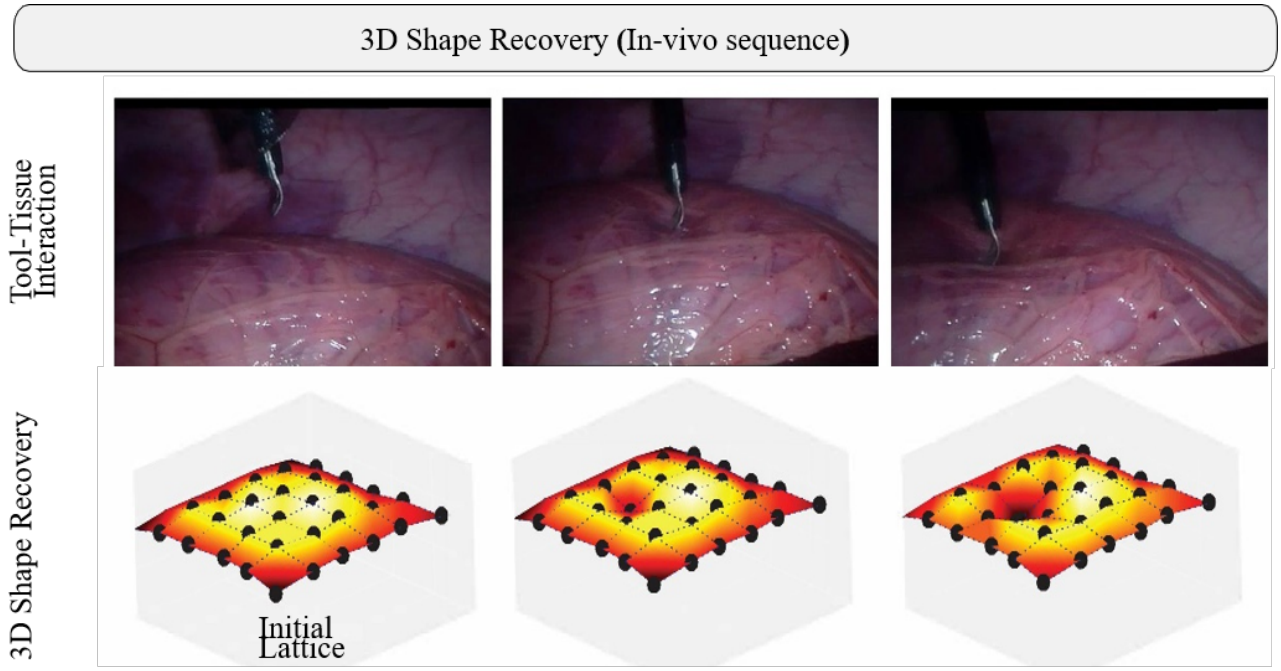

Figure 9: 3D lattice computation from the surface reconstruction.

- Implementation of the algorithm: The algorithm has been implemented in MATLAB to ease the development process and we are currently translating it to C++ so as it can be integrated in the global system.
- Test and validations: This MATLAB algorithm implementation has been tested and validated.

Force Estimation As previously mentioned, our strategy to compute force feedback relies on using a Recurrent Neural Network (RNN) to get the relationship between the deformation and the applied force. The input to this neural network are the lattice parameters and the geometric information. Specifically, we propose the usage of a Long-Short Term Memory (LSTM) based architecture to compute the force feedback.

The work done in this part of the problem is:

- Development of the LSTM-RNN: The architecture used combines an architecture based on LSTM-RNN with two types of hidden layers: with basic units (layer 1) and cells (layer 2). These cells are composed of a set of units that enforce constant error flow which helps stabilizing force estimation. The LSTM-RNN has been implemented in MATLAB and we are currently translating the code to C++ to integrate this system to the global project.
- Training: A set of different silicon models with shores reproducing different tissues dynamic properties has been used to obtain data to train the LSTM-RNN. To obtain the data a robot

with a calibrated force sensor at its waist and a surgical tool were used (Fig. 10). The same data will be used to train the new LSTM-RNN in the new version implemented in C++.

- Test and Validation: The LSTM-RNN has been tested and validated using the preliminary MATLAB version.

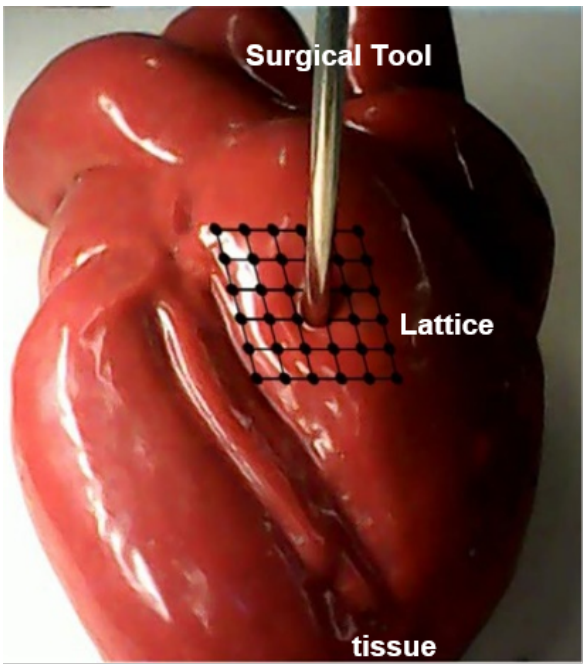

Figure 10: Example of silicon model used in the training.

Integration to the SARAS platform A ROS node has been implemented for the integration of the force estimation module to be used to provide force feedback in the SARAS platform. This node will need the images and position of the camera from the vision system and in return, it will provide a 6-component vector (3 for the force vector and 3 for the torques) in a ROS topic. In this way, the force estimation module will be a black box, acting as a simple sensor, for SARAS. The integration to SARAS will involve two main tasks, a basic approach where the LSTM-RNN will be trained using the tissues that will be used in the procedures and the definition of the communication protocols and acquisition frequency of the images from the camera to the ROS node.

2.2 Augmented reality for visual feedback

In standard laparoscopic operation, the assistant surgeon operates coordinately with the main surgeon monitoring the operating area by means of a high resolution 3D display placed in the operating room, which shows the images captured by the endoscope. In order to increase accuracy and correctly compute proceedings, in the SARAS project the assistant surgeon will be equipped with a 3D Head-mounted display, enhancing the visual feedback. The natural straight forward implementation of visual feedback is display the same images provided to the standard 3D display inside 3D Head-mounted display. The main problems related to the transport of images from the 3D display to the 3D Head-mounted display are:

- Interfacing the new hardware and software.
- Processing the images in order to provide the user the 3D perception of the operating area.

2.2.1 Stereo imaging

A well-known technique for making or enhancing the illusion of depth, or in general the 3D perception, is the stereo imaging (also called stereoscopy). This technique is based on provide two offset images separately to the left and right eye of the viewer. These images are then automatically elaborated by the brain to give the 3D perception. The 3D head mounted display is made-up with two internal small LCD displays with magnifying lenses, one for each eye, independently controlled. The endoscope of the Da Vinci robot equip an analogic stereo camera that is used to stream the images during the surgical operation. Since the stereo camera returns the left and the right channel images, the implementation of visual feedback can be easy achieved streaming separately the two images into the two LCD display of the viewer, opportunely processed to fit with the displays requirement. Using this approach, it is also possible to develop image processing algorithms considering as input only one two-dimensional image and then apply the same concept to both right and left channel images to keep the 3D perception.

2.2.2 Interfacing hardware and software

A Oculus Rift 3D Head-mounted display with two motion tracking camera interfaced with a Windows 10 Desktop PC will be used to provide the images to the assistant surgeon while the raw images are provided by the endoscope of the Da Vinci robot research kit, correctly interfaced with an Ubuntu Desktop PC. The different PCs are connected between each other by means of an Ethernet switch which provide the communication. On the Ubuntu side, the ROS (Robot Operative System) framework is used to implement the endoscope software management and to create a web-socket server for the communication between PCs. On the Windows side, Unity3D software is used to connect as a client to the web-socket, extract the images streamed by the endoscope through the web-socket and provide the images to the Oculus Rift. A schematic overview of the setup is reported in Fig. 11 In order to provide images to the viewer, a Unity scene was developed to implement the stereo imaging. Once interfaced with the web-socket, which provides independently the right and the left stereoscopic images, two different planes in the Unity environment are used to show the input images. Two different cameras are than located in front of the planes and stream the images independently to the right and left displays of the viewer. The relative location between the camera and the corresponding plane allow to adjust the size of the projected image. A screenshot of the scene implemented in Unity is reported in Fig. 12, while in Fig. 13 a picture with the user making use of the visual system is reported.

2.2.3 Remarks

The main problems related to this approach are due to the so called virtual reality sickness: when expose to a virtual environment, user's perception of self-motion is based on incongruent sensory inputs from the visual system, vestibular system, and non-vestibular proprioceptors. When these inputs are at odds with the user's expectation sickness can occurs in the form of discomfort, headache

Figure 11: Schematic overview of the setup for augmented reality.

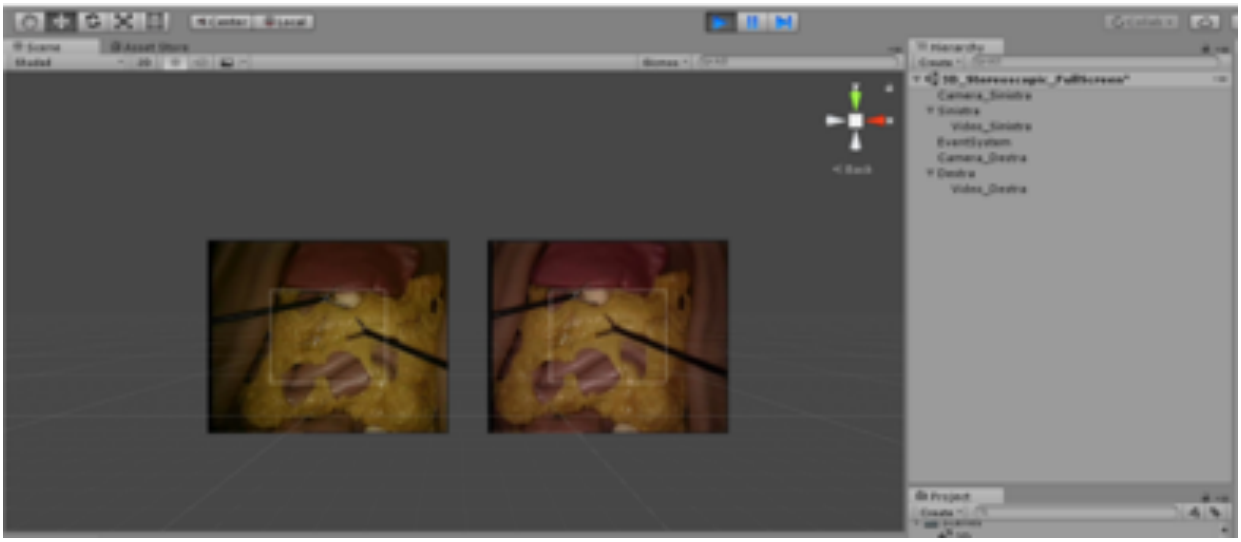

Figure 12: Scene implemented in Unity.

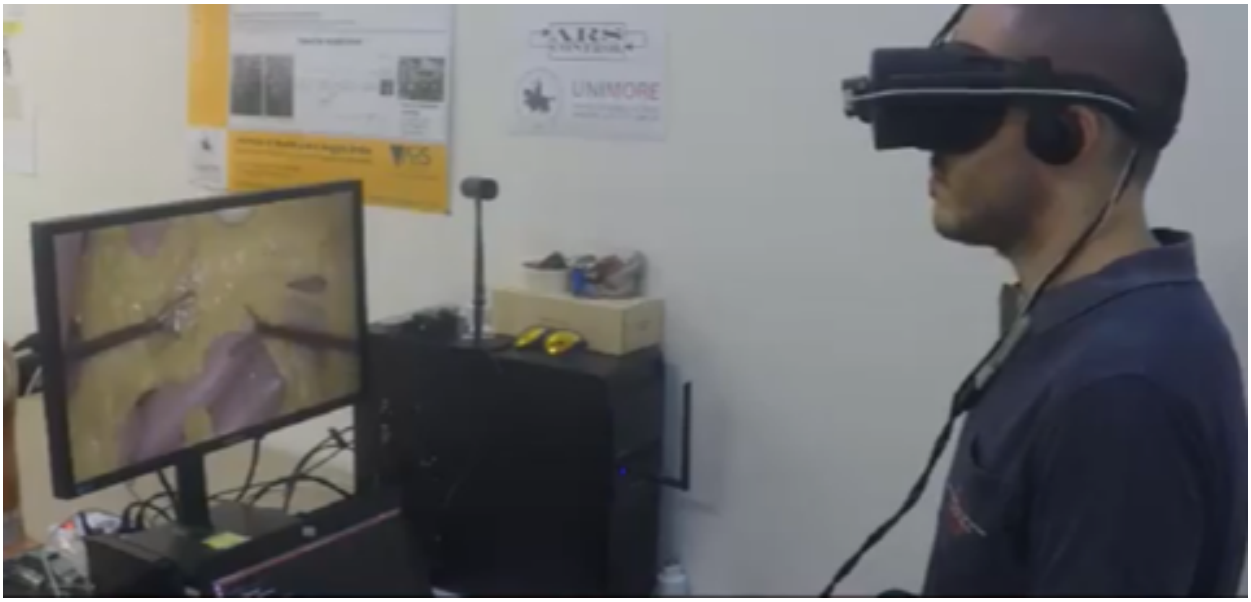

Figure 13: User experience with the visual system.

and stomach awareness. Since the images streamed to the viewer will be fixed with the user motion and the endoscope, which cause the change of the view, is moved by the main surgeon, sensory conflict can occur. Considering that the movements of the assistant surgeon view are very closed to the operating area and the movements of the endoscope are slow and not sudden, the strategy approached can be consider reliable with respect to these problems

2.3 Colliding regions identification

The identification of the regions where the collisions between tools and obstacles can occur is an useful support for ensuring safety during the operations: in multi-master/multi-slave bilateral teleoperation platform, in fact, it can provide to the surgeons information that could be hard to obtain from the images while in the autonomous platforms it can provide information that are necessary for planning choices and movements properly. The following sections will describe what the identification of a colliding region is and how this is implemented in SARAS platforms.

2.3.1 Colliding region definition

The colliding region of a tool at a certain time can be defined as the union of the intersections of the volume occupied by the tool and the volumes occupied by the so-called "forbidden regions", which are simply the obstacles (in the laparoscopic surgery scenario they can be the other tools, the organs and the pelvic bones) :

$$C_T(t) = \bigcup_{i=1:n} \left(T(t) \bigcap F_i(t) \right) \tag{42}$$

where $C_T(t)$ is the colliding region at time t, $T(t)$ is the volume occupied by the tool at time t and $F_i(t)$ is the volume occupied by the $i-th$ forbidden region at time t.

From this, the computation of the colliding region for a moving tool in a certain time interval can be done by computing the union of the colliding regions of the tool at each time of the interval.

$$C_T = \bigcup_{t \in [t_{init}, t_{final}]} (C_T(t)) \tag{43}$$

From these equations it follows that if the motion is collision free, the colliding region of the tool is a void set. Equation (42) and (43) can be used to foretell future colliding regions of an evolving systems: in fact, if all the motions in the scenario are known in a certain time horizon,also the volumes occupied by the elements are known so the the equations are applicable.

2.3.2 Colliding region identification for multi-master/multi-slave platform

The principal aim of the colliding region identification for the multi-master/multi-slave platform is to give the surgeons additional visual feedback to help them understand the scenario in which they are operating and so improving the safety of the actions. In this case one the most important thing to take in count when developing the colliding region identification algorithm is that the information provided must be useful and easy to understand.

Recalling (42) and (43), it is clear that the computed colliding region can have non trivial shapes; moreover, considering that a tool can not continue with its original motion after the first collision occurs, the colliding region computed following the definition above can include spatial regions that are not physically reachable. From these considerations it is easy to understand that giving the whole colliding region as a visual feedback to the user is not a good choice. As a consequence, also its whole computation can be useless. Moreover, in this platform the user is mainly interested in understanding how far the tool is from a possible collision or to distinguish if a force feedback is due to the teleoperation or to a collision with an obstacle. For all these reasons,we decided that computing the whole colliding region is not needed.

Finally, since the tools are driven by the user, and the knowledge on how the tool will move resides only in the users, it is hard to design a meaningful time horizon for the movement prediction. In fact, a too large time horizon brings misleading information while a too short time horizon will not give more useful information than a static colliding region computation.

For all these reasons, for this platform the colliding region identification problem can be simplified into a static collision checking between the volumes of the tools and the forbidden regions: this will still give the user a very useful information without overloading him with useless feedback.

The algorithm implemented performs a cyclical distance computation checking all the distances between the tools and the forbidden regions and keeping for each tool the minimum of such distances (i.e. the distance from the closest object) as shown in pseudocode of algorithm 1).

In order to perform a fast computation of the distances, each tool and each forbidden region is wrapped with a bounding volume of well-known shapes: this allows to use ad-hoc fast distance computations algorithms. The laparoscopic tools are modeled as capsules, which are cylinders with hemispherical terminations; the end-points of the axis of the capsule are the tool-tip and the trocar while the radius is the simply the radius of the tool at its wider point(see figure 14). In figure 14

Algorithm 1 Distance_computations

```
tools={daVinci1,daVinci2,assistantTool1,assistantTool2 };
forbidden_regions=[parallelepiped1 ... parallelepipedN];
distances_daVinci1=[];
distances_daVinci2=[];
distances_assistantTool1=[];
distances_assistantTool2=[];
while cycling==true do
  update_pose(tools);
  d=check_distance(daVinci1,daVinci2);
  append(d,distances_daVinci1);
  append(d,distances_daVinci2);
  d=check_distance(daVinci1,assistantTool1);
  append(d,distances_daVinci1);
  append(d,distances_assistantTool1);
  d=check_distance(daVinci1,assistantTool2);
  append(d,distances_daVinci1);
  append(d,distances_assistantTool2);
  d=check_distance(daVinci2,assistantTool1);
  append(d,distances_daVinci2);
  append(d,distances_assistantTool1);
  d=check_distance(daVinci2,assistantTool2);
  append(d,distances_daVinci2);
  append(d,distances_assistantTool2);
  d=check_distance(assistantTool1,assistantTool2);
  append(d,distances_assistantTool1);
  append(d,distances_assistantTool2);
  for each f in forbidden_region do
    d=check_distance(daVinci1,f);
    append(d,distances_daVinci1);
    d=check_distance(daVinci2,f);
    append(d,distances_daVinci2);
    d=check_distance(assistantTool1,f);
    append(d,distances_assistantTool1);
    d=check_distance(assistantTool2,f);
    append(d,distances_assistantTool2);
  end for
  min_dist_daVinci1=min(distances_daVinci1);
  min_dist_daVinci2=min(distances_daVinci2);
  min_dist_assistantTool1=min(distances_assistantTool1);
  min_dist_assistantTool2=min(distances_assistantTool2);
end while
```

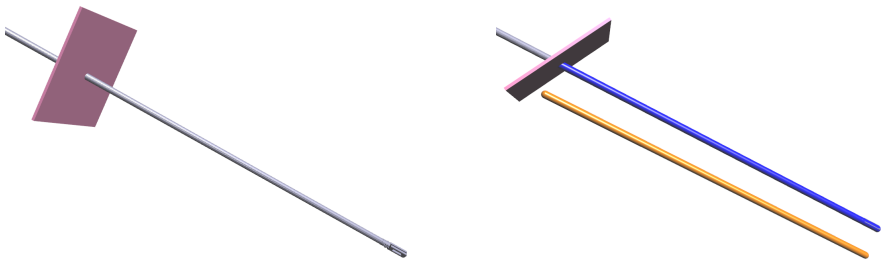

Figure 14: Model of the instrument and its related capsule

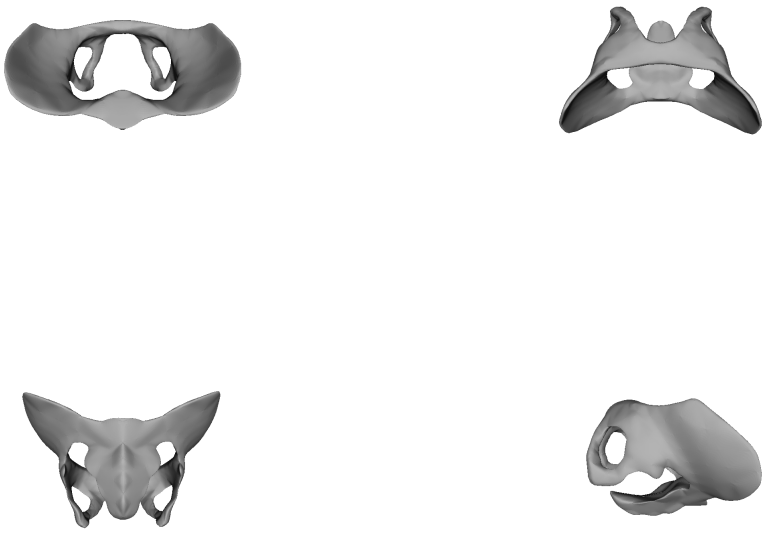

Figure 15: Views of the 3D model of the pelvic bones

the pink plane represents the trocar, the orange solid is the capsule used to model the instrument while the blue model is simply the orange capsule in its final position (bounding the instrument).

The forbidden region of the pelvic bones is modeled with a compositions of parallelepipeds starting from the 3D model of the bones (see figure 15 and 16).

As shown in figure 16, the parallelepipeds are inserted only in the internal part of the pelvic cavity: that is because the other parts of the pelvic bones are outside the robot's workspaces; moreover, it is possible to notice that the lower part of the cavity is not modeled with parallelepipeds. Indeed, that zone is not interested in colliding region computation, since it is where the organs reside. The organs are not considered as forbidden regions for this platform because it is possible for the surgeons to interact with them during the operation. The choice of capsules and parallelepipeds allows to reduce the capsule-to-capsule and capsule-to-parallelepiped distance problems to simpler

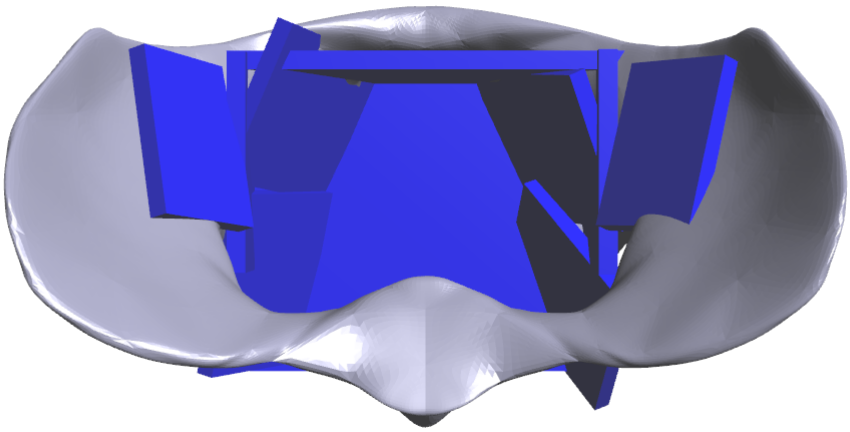

Figure 16: Bounding boxes for the pelvic bones

segment-to-segment and segment-to-plane problems.

The segment to segment distance computation problem is solved using the algorithm proposed in the article [14]: briefly, the segments are properly parameterized with t and u parameters, then simple expressions coming from a distance minimization problem between lines are used to compute t and u and their values are checked and bounded to be in the $[0, 1]$ parameterization interval; t is computed and bounded again and finally using t and u the minimum distance is computed. This algorithm is used in a capsule-to-capsule distance computation simply considering that the axes of the capsule are two segments and that the minimum distance between two capsules is the distance between the two axes minus the radii of the capsules, since each point on the surface of a capsule has a distance to the axis equal to the radius.

Capsule-to-parallelepiped minimum distance can be found looking for the minimum distance between the capsule and all the edges and the faces of the parallelepiped; since an edge can be considered as a capsule with the radius equal to 0, the 12 computations for the capsule-to-edge distances can be performed using the algorithm already mentioned. Following the same principle already explained (i.e. taking count of the capsule radius afterwards), the capsule-to-faces problem can be reduced to segment-to-faces problem; this again simplifies the problem,since the minimum distance can be determined by checking only the two end-points,in fact:

- if both the projections of the end-points on the plane where the nearest face lies are internal to the face, the minimum distanpoints could be useful for future implementation as they can be as application points for force feedback in the future implementations.ce is determined by one of the end-points;
- if one or both projections are external to the face, the minimum distance can actually be determined by a point on the axis which is not one of the end-point, but then the minimum

distance is for sure between the axis and one edge of the face, so checking only the end-points will produce a distance value which will not be minimum and so it will be discarded when the distance computation with the edges will be performed.

The distance between an end-point and the faces can be simply and efficiently computed aligning the end-point and the faces to a conventional reference frame and properly checking the maximum and minimum values of the coordinates.

Algorithm 2 Capsule_to_parallelepiped_distance_computation

```
get_point_to_faces_distance(faces,end_point_1);
get_point_to_faces_distance(faces,end_point_2);
for each e in edge do
  capsule_to_capsule_distance(capsule,e,capsule.radius,0);
end for
keep_minimum_distance(t);
```

As a final remark, both of the functions implemented can give some information about how much two objects are intersecting by returning a negative distance value; this can be an useful information for the estimation of the force that should be given as a feedback during the interaction between the objects. Moreover, these functions are also capable of computing the points on the object's surfaces which are at minimum distance (when they exists and are unique): these points could be used as references for the computation of directed assistive forces (i.e. virtual fixtures for collision avoidance) in future enhancements of the bilateral teleoperation scheme.

2.3.3 Colliding region identification for future autonomous platforms

For the development of the future autonomous platforms the proper colliding region computation concepts of equation (42) will be taken in count: this because the dimension and the shape of the colliding region could give useful information for the computation and the optimization of a collision free autonomous movement. Moreover, since all the phases of the operation will be planned it will be possible to properly use equation (43) to foresee the colliding region; we also expect to take into account uncertainties and safety margins during the computation of the volume occupied by the tool during the movement in order to obtain more reliable results.

3 Experimental validation of the SARAS MULTIROBOTS-SURGERY Platform

In this section the communication and control structure for regulating the interaction between the main surgeon and the assistant surgeon in the MULTIROBOTS-SURGERY platform is described. The main surgeon and the assistant surgeon operate their robots in a shared environment and, therefore, a strategy for helping the surgeons to avoid collisions during the operation has been developed. Given the kinematics of the robots, the distances between the laparoscopic tools and the forbidden regions are computed, so the possible colliding region is identified . Indicators for the collisions and the distances are displayed both on the da Vinci console and on the Oculus Rift worn by the assistant surgeon. Multi-robot control strategies have been merged with the teleoperation architecture to guarantee a safe interaction of the laparoscopic tools.

3.1 Description of the multi-master/multi-slave bilateral teleoperation architecture

The SARAS experimental setup is shown in Figure 17:

- Figure 17.(a) shows the SARAS assistant console where the Simball device is connected to two Touch haptic devices for rendering the force feedback. The Oculus Rift allows the assistant to have the same view of the main surgeon at the da Vinci console.
- Figure 17.(b) shows the SARAS slave arms holding two laparoscopic tools: the left arm a grasper and the right arm a forcep. In the same figure it is possible to see the da Vinci arms and the endoscope teleoperated by the main surgeon.
- Figures 17.(c) and 17.(d) show the overall SARAS setup. On the bottom-right it is possible to see the da Vinci console.

Even though the phantom in Figure 17.(b) is without the peritoneum to better understand the interaction and the relative pose of the robotic arms, the positions of the SARAS arms have been discussed in details with the OSR surgeons and chose as a trade-off between:

- workspace needed to perform the surgical tasks, and
- mechanical constraints due to the size of the da Vinci arms, SARAS arms and trocars on the peritoneum.

In real tests of prostatectomy (RARP) and nephrectomy procedures, the configuration of the trocars for the robotic arms will be the ones shown in Figure 18 as explained in D1.1 – Requirements for surgical actions. An important advantage of the SARAS platform is that it is possible to locate the laparoscopic tools on both side of the operating table. Such configuration is not possible in standard robotic MIS where the assistant is either on the right or left side of the patient.

The SARAS MULTIROBOTS-SURGERY experimental platform is fully described in D7.1 – Technical Specifications and D7.2 – Software/Hardware architecture for the MULTIROBOTS-SURGERY

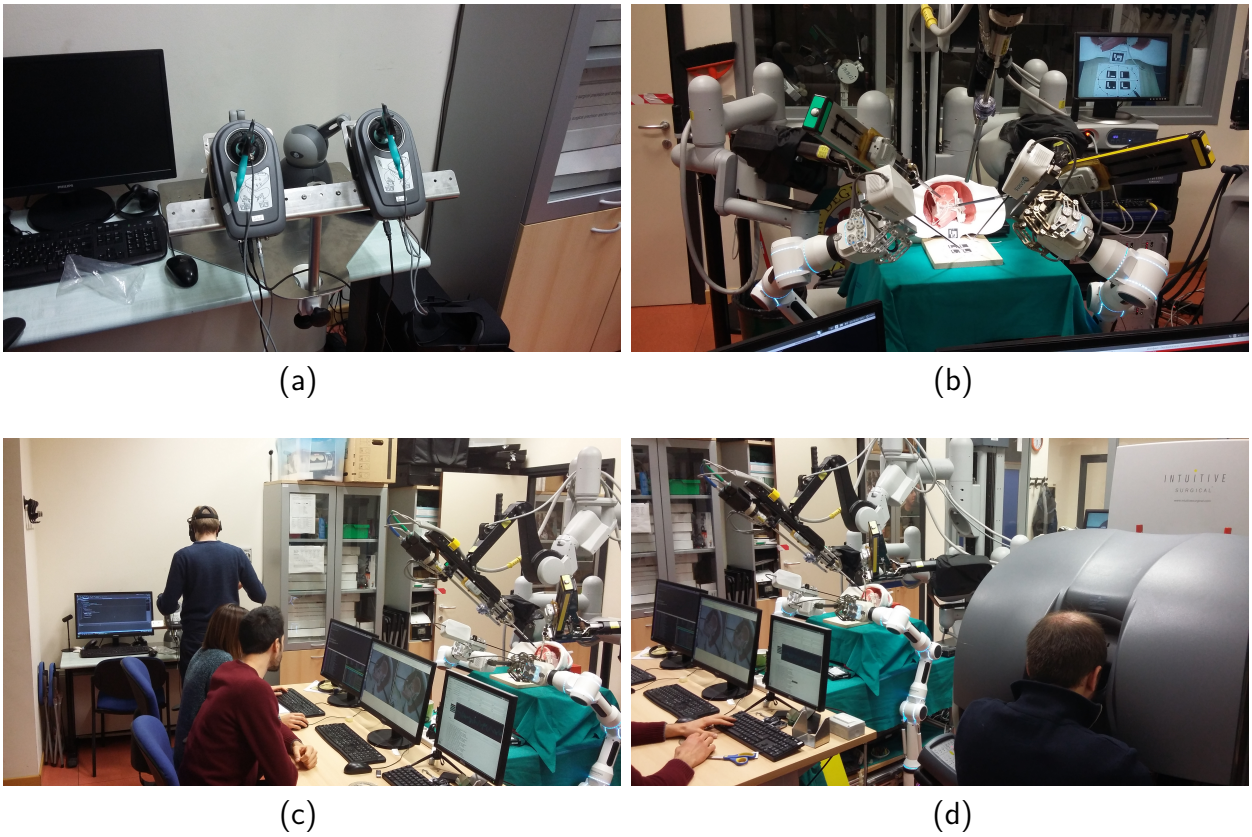

Figure 17: The SARAS MULTIROBOTS-SURGERY experimental setup.

platform: in Figure 19 we just recall the overall architecture. The yellow *Teleoperation module* is composed of two software modules

- the *Assistant console*: it interacts with the Touch haptics devices and the Simball for the force rendering and with the Ocolus Rift for super-imposing colors when collisions between tools and anatomical structures are predicted in the next future, and
- the *Assistant controller*, where the teleoperation algorithm is implemented together with the Colliding regions identification algorithm.

They are ROS components that exchange data with the DVRK and the *ROS core and Data recording* as shown in Figure 19.

3.2 Validation on the teleoperation architecture

The proposed teleoperation architecture was implemented and validated at UNIVR using the SARAS hardware/software setup. With respect to the preliminary validation performed at UNIMORE, the two teleoperated SARAS arms at the slave side are not equipped with force sensors. For this reason the teleoperation architecture was translated from position-force control to position-position control and no forced delays were introduced in the system.

Experiments were performed to check the correct behavior of the teleoperation architecture using the SARAS setup described in the previous subsection. Experimental results are reported in the

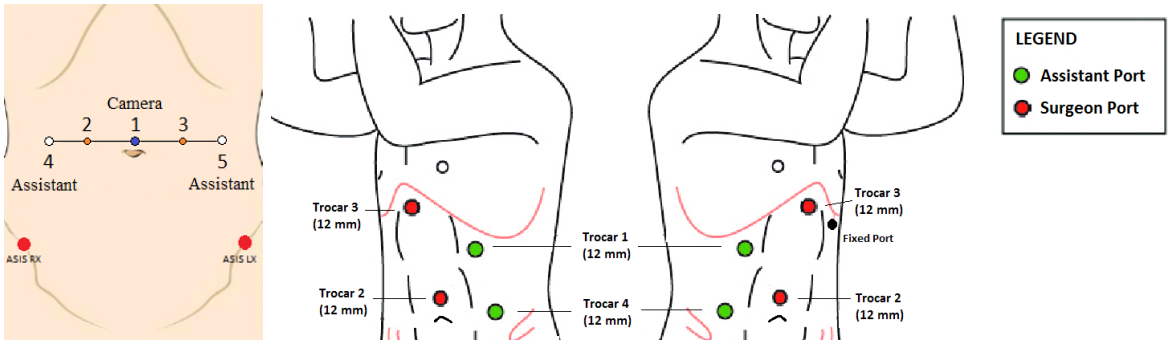

Figure 18: Trocars' position for prostatectomy (left) and nephrectomy (right).

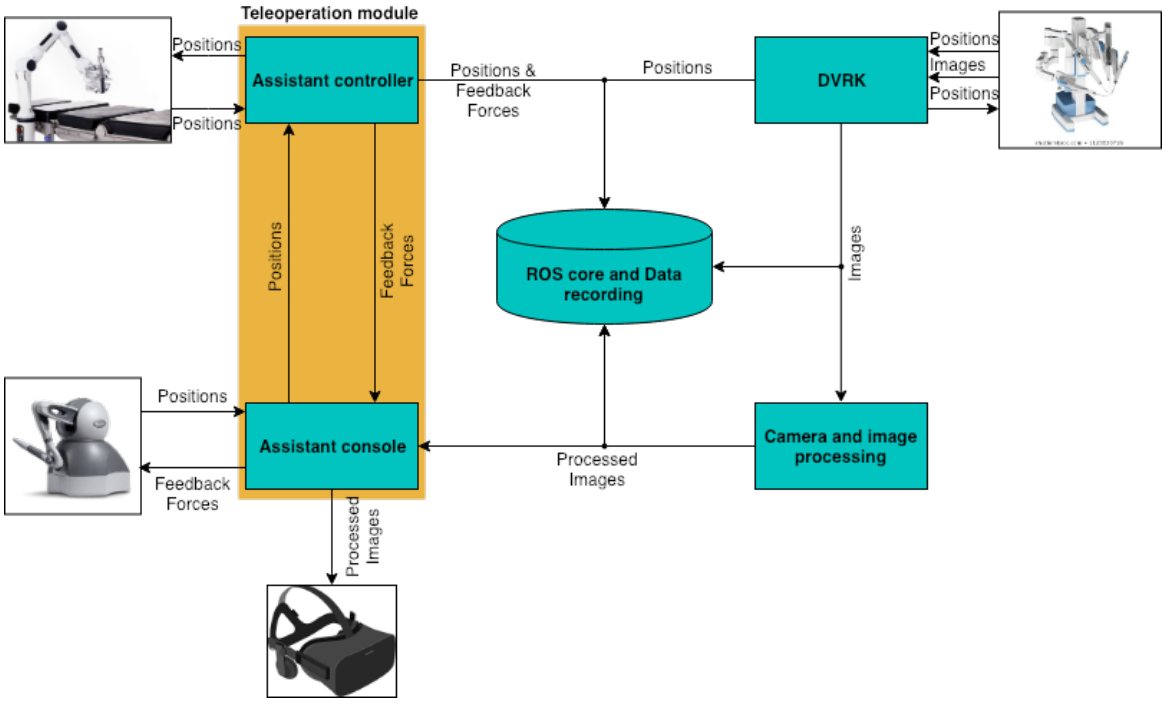

Figure 19: MULTIROBOTS-SURGERY platform: Hardware and Software.

following figures. Figure 20 shows the position along the x axis for the two master-slave pairs (indicated with *right* and *left* in the plots).

The position measurements show how the slave robots holding the laparoscopic tools follow the master robots handled by the operator. The delays between the two sides are introduced by the software architecture connecting the teleoperation module with the low-level controllers. This delay will be shortened soon with new software updates.

The position mismatch between the master device and the slave device can be felt by the user thanks to the force feedback provided by the haptic devices at the master console. The virtual springs placed between the end-effector of the slave robots and the tips of the master instruments produce an elastic force that tries to align the two positions making the user feel that the system is moving. Figure 21 shows the (left and right) forces along the x axis generated by the Touch haptic devices connected to the Simball handles. Comparing Figures 20 and 21 it is easy to see that the larger the position displacement, the larger the value of the force.

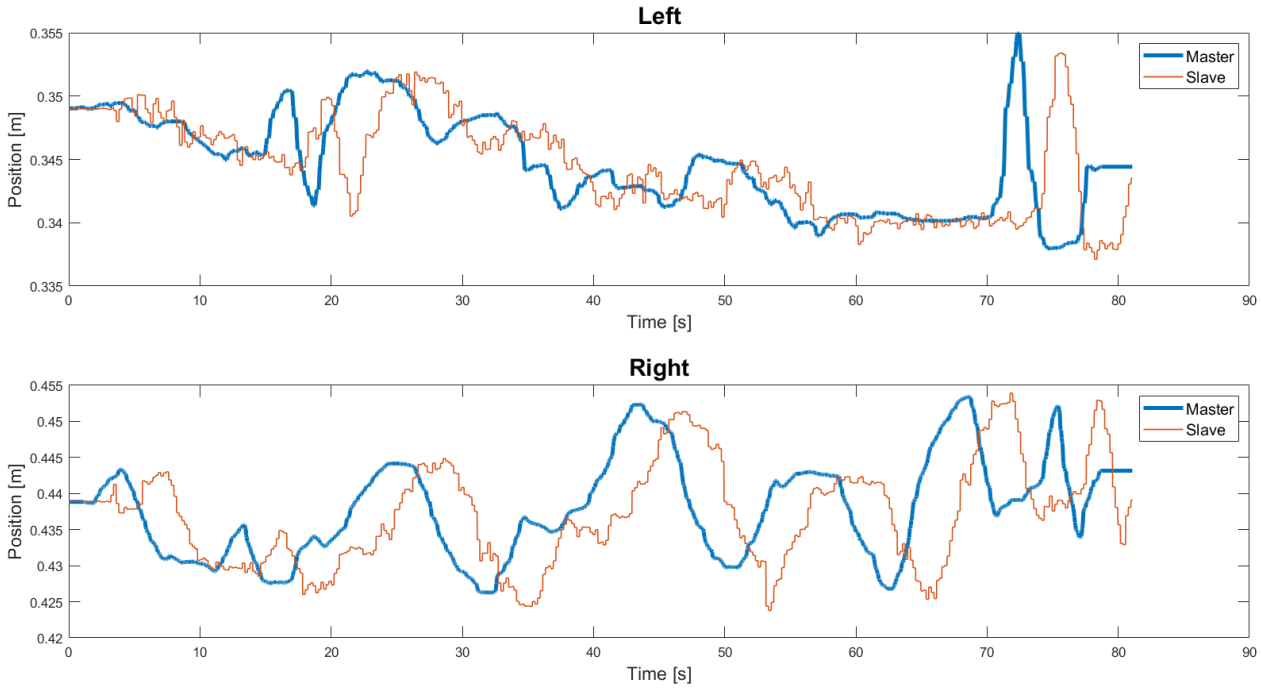

Figure 20: Cartesian position of the master devices (blue line) and of the slave device (orange line) along the x axis, for the left side and the right side.

Despite of the delay introduced by the software architecture, the overall behavior of the MULTIROBOTS-SURGERY platform is stable and the tracking performance are good. The virtual coupling between the master and the slave allows the user to perceive the motion of the slave and the amount of the position mismatch.

3.3 Colliding regions displaying

The goal is to provide the least intrusive yet most informative feedback regarding collision events to the surgeon as they interact with the SARAS manipulators during the tasks. For this reason, the system provides a visual feedback system that offers at the same complete and unobstructed view of the operative region and a reduced cognitive overload on the user. This is achieved with an augmented reality projection of information on the shafts of both laparoscopy tools. The system uses simple colored lines with the following color convention to relay information:

[no color] the instruments are at a safe distance from any collision; the surgeon does not receive unnecessary information;

- [yellow] the instruments are operating at a close distance to a collision event;
- [blue] the instruments are colliding with either themselves or a forbidden region.

Such information follows the instruments' pose step-by-step, thus giving the user an immediate feedback on the status of the tools with respect to the environment.

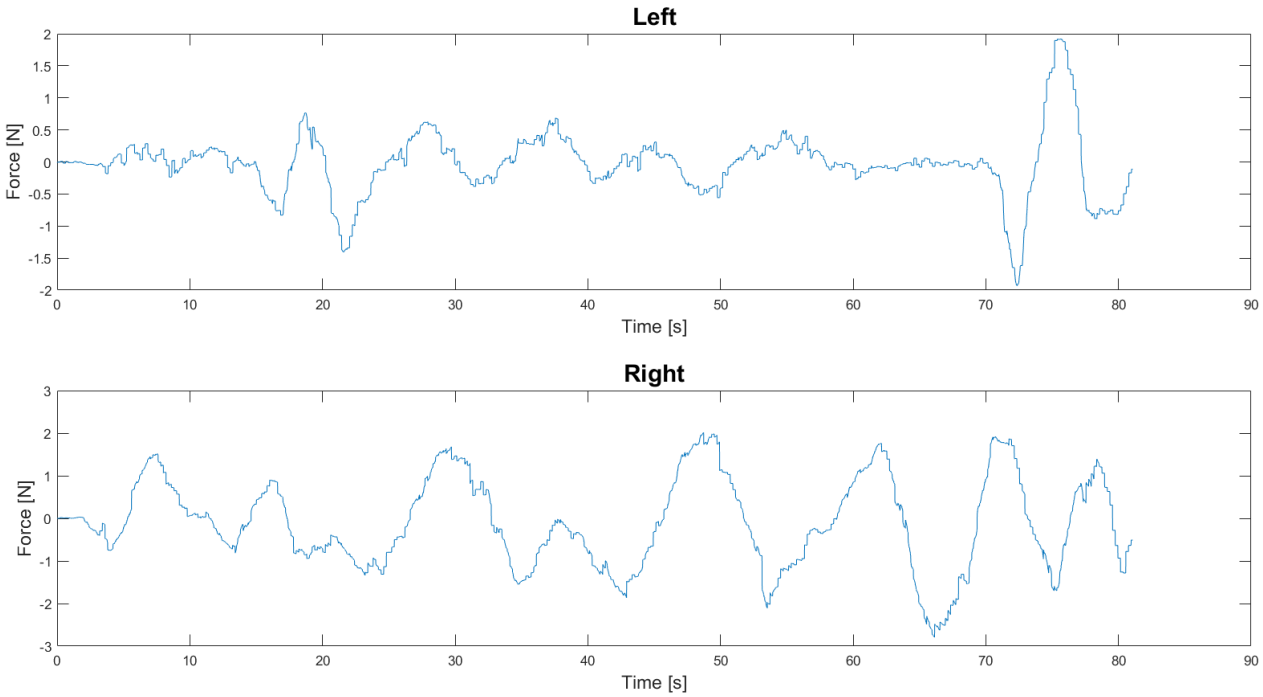

Figure 21: Forces applied at the master devices

The lines are drawn in the operational space seen by the stereo camera and, then, projected in both left and right images using the standard pinhole model approximation; this generates a three dimensional augmented reality view over the operative region.

The pinhole model employs the knowledge of the camera intrinsic and extrinsic parameters acquired during the calibration phase to project points in the 3D space to the 2D image plane (Figure 22): given the image principal point (c_x, c_y), the optical axis $\mathbf{z}_c$, and the focal point $\mathcal{F}_c$, it is possible to project the 3D point $P = (X, Y, Z)$ on the point $p\prime = (u, v)$ of the image plane; this is possible through the equation $p\prime = K[R|t]P$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{44}$$

where K is the intrinsic camera matrix and $[R|t]$ is the rotation/translation extrinsic matrix.

Information regarding collision events is provided by the collision detection node (Section Colliding Regions Identification) as a list of shortest distances between tools and the environment.

A run-time multi-threaded mapping between collision information, left and right camera images, and color coding, allows the visualization node to operate seamlessly with all other software components.

The 3D geometrical shapes that comprise both the forbidden and tool colliding regions are hidden from virtual reality headset for the user to see their location within the operative space. This projection requires the headset to be mapped to the stereo camera Cartesian frame for the surgeon's head motions to correspond to the view projected on both screens. A set of experiments to validate the proposed colliding regions algorithm, which is presented in Section 2.1.1, was conducted on the

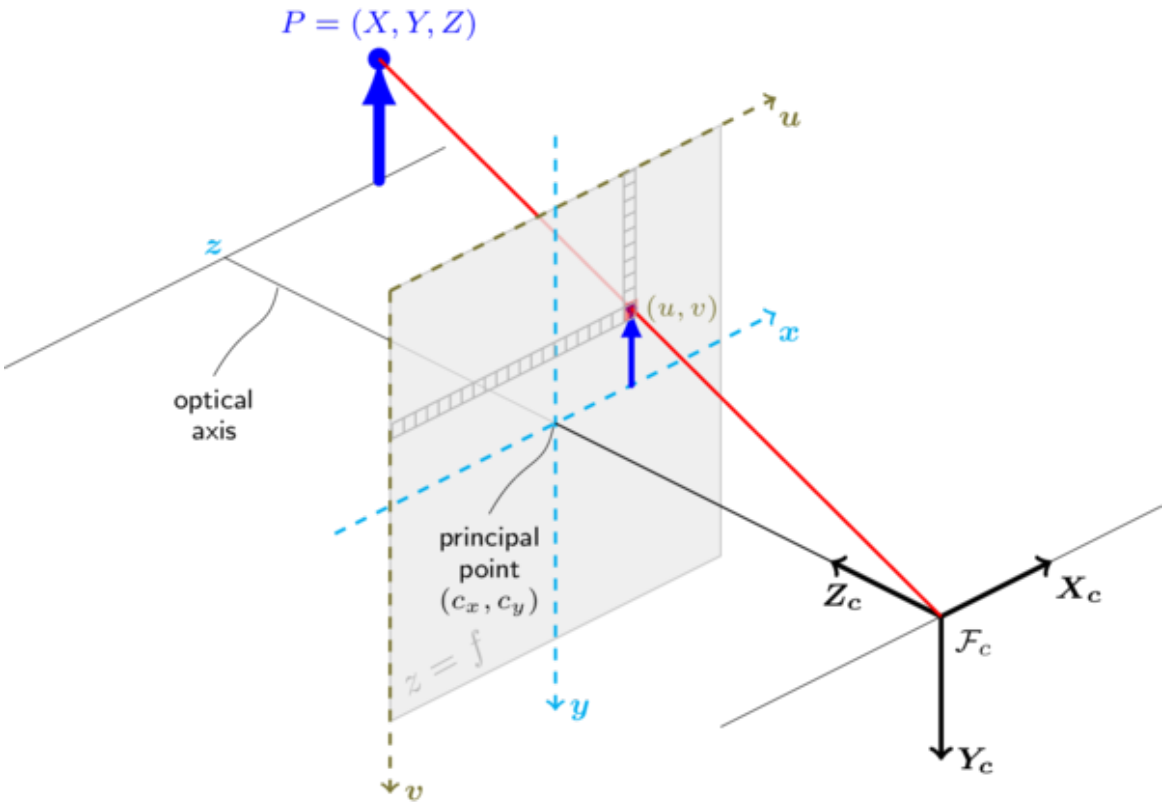

Figure 22: Pinhole camera model projection. From `docs.opencv.org`

SARAS pelvic region phantom. The evaluation strategy relies on the augmented reality collision visualization to verify the implementation soundness.

The color-coded visual feedback acts in synergy with the collision detection node to provide user notifications for potential and imminent danger to the patient, without neither obstructing nor overloading the surgeon's view of the operative region since information is depicted directly on top of each shaft position. When all motions are nominal, no information is required from the colliding regions so no line is pictured on the tools (Figure 23); when either instrument gets close to a collision event (with either another instrument or a forbidden region), the yellow line warns about the eventuality of a collision for the time the user maintains the tool within range (Figure 24); finally, when the instruments enter a predetermined collision boundary which is still a few millimeters away from an actual collision, the blue line indicates an imminent danger to the system and/or the patient (Figure 25). When either instrument moves outside the view of the endoscope, a single colored dot at the image border maintains contact with the position of the tool outside the view of the surgeon, offering useful additional feedback on the system's status.

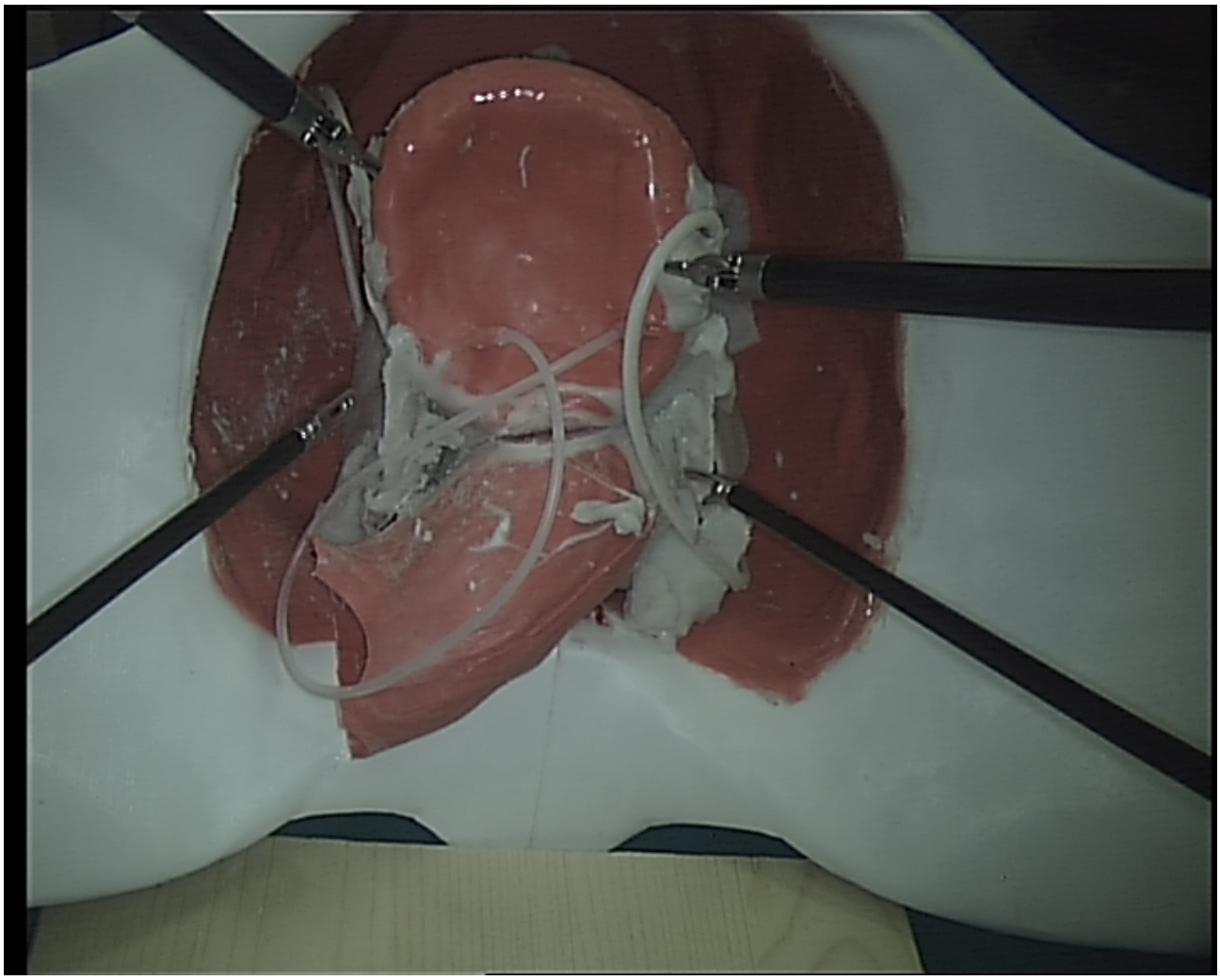

Figure 23: Testing of collision-safe motion: the two SARAS tools are located at a safe distance from both tools and forbidden regions; no information is provided to avoid possible cognitive overload.

References

[1] A. I. Aviles, S. M. Alsaleh, J. K. Hahn, and A. Casals. Towards retrieving force feedback in robotic-assisted surgery: A supervised neuro-recurrent-vision approach. *IEEE Transactions on Haptics*, 10(3):431–443, July 2017.

[2] B. Bayle, M. Joinie-Maurin, L. Barbe, J. Gangloff, and M. de Mathelin. Robot interaction control in medicine and surgery: Original results and open problems. *Book Chapter in Computational Surgery and Dual Training*, pages 191–196, 2014.

[3] O. Van der Meijden and M. Schijven. The value of haptic feedback in conventional and robot-assisted minimal invasive surgery and virtual reality training: a current review. *The International Journal of Robotics Research*, 34(14):1773–1787, 2015.

[4] M. Diana and J. Marescaux. Robotic surgery. *British Journal of Surgery*, pages 15–28, 2015.

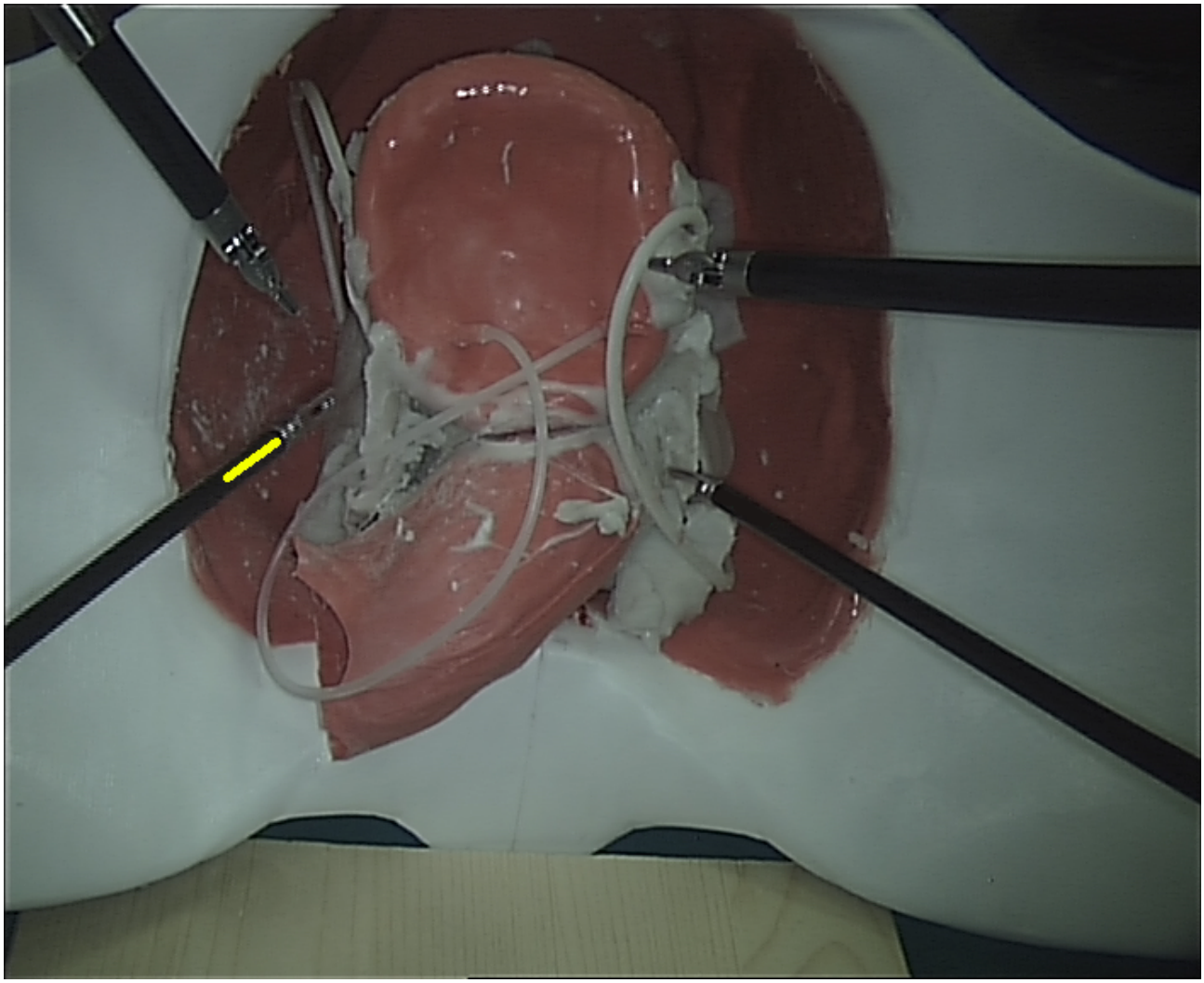

Figure 24: Testing of collision warning feedback: the left SARAS tool is within range of the left daVinci, thus triggering a yellow warning line on the tool.

[5] Vincent Duindam and Stefano Stramigioli. Port-based asymptotic curve tracking for mechanical systems. *European Journal of Control*, 10(5):411–420, 2004.

[6] A. Faragasso, J. Bimbo, Y. Noh, A. Jiang, S. Sareh, H. Liu, T. Nanayakkara, H. A. Wurdemann, and K. Althoefer. Novel uniaxial force sensor based on visual information for minimally invasive surgery. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1405–1410, May 2014.

[7] Federica Ferraguti, Nicola Preda, Marcello Bonfe, and Cristian Secchi. Bilateral teleoperation of a dual arms surgical robot with passive virtual fixtures generation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4223–4228, 2015.

[8] Federica Ferraguti, Cristian Secchi, and Cesare Fantuzzi. A tank-based approach to impedance control with variable stiffness. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4948–4953, 2013.

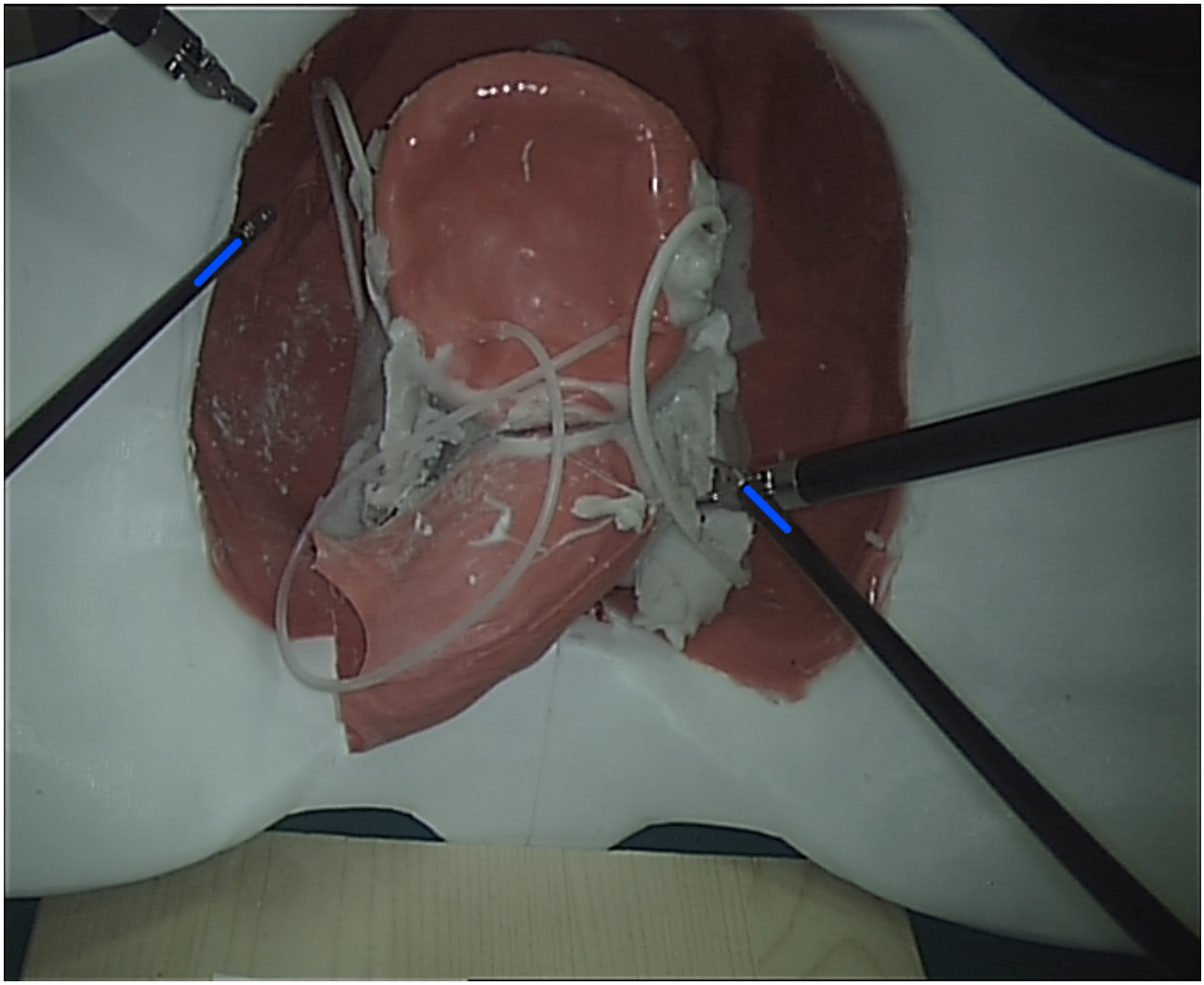

Figure 25: Testing of ongoing collision feedback: the right SARAS tool is in direct contact with the tool while the left is in collision with the pelvic bone colliding region; both correctly display a blue collision warning line.

[9] Antonio Franchi, Cristian Secchi, Hyoung Il Son, Heinrich H Bulthoff, and Paolo Robuffo Giordano. Bilateral teleoperation of groups of mobile robots with time-varying topology. *IEEE Transactions on Robotics*, 28(5):1019–1033, 2012.

[10] Michel Franken, Stefano Stramigioli, Sarthak Misra, Cristian Secchi, and Alessandro Macchelli. Bilateral telemanipulation with time delays: A two-layer approach combining passivity and transparency. *IEEE transactions on robotics*, 27(4):741–756, 2011.

[11] Kristen L. Helton, Buddy D. Ratner, and Natalie A. Wisniewski. Biomechanics of the sensor-tissue interface - effects of motion, pressure, and design on sensor performance and foreign body response - part ii: Examples and application. *Journal of Diabetes Science and Technology*, 5(3):647–656, 2011.

[12] Dongjun Lee and Ke Huang. Passive-set-position-modulation framework for interactive robotic systems. *IEEE Transactions on Robotics*, 26(2):354–369, 2010.

[13] T. Lendvay, B. Hannaford, and R. Satava. Future of robotic surgery. *The Cancer Journal*, 19(2):109–119, 2013.

[14] Vladimir J. LUMELSKY. On fast computation of distance between line segments. pages 55–61, 1985.

[15] A.M. Okamura, L.N. Verner, C.E. Reiley, and M. Mahvash. Haptic feedback in robot-assisted minimally invasive surgery. In *Book Chapter in Robotics Research*. Springer Tracts in Advanced Robotics, 2011.

[16] C. Pacchierotti, L. Meli, F. Chinello, M. Malvezzi, and D. Pratichizzo. Cutaneous haptic feedback to ensure the stability of robotic teleoperation systems. *British Journal of Surgery*, pages 15–28, 2015.

[17] C. Pacchierotti, A. Tirmizi, and D. Prattichizzo. Improving transparency in teleoperation by means of cutaneous tactile force feedback. *ACM Transactions on Applied Perception*, 2014.

[18] M. Pizzoli, C. Forster, and D. Scaramuzza. Remode: Probabilistic, monocular dense reconstruction in real time. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[19] P. Puangmali, H. Liu, L. D. Seneviratne, P. Dasgupta, and K. Althoefer. Miniature 3-axis distal force sensor for minimally invasive surgical palpation. *IEEE/ASME Transactions on Mechatronics*, 17(4):646–656, Aug 2012.

[20] S. Sokhanvar, J. Dargahi, S. Najarian, and S. Arbatani. Clinical and regulatory challenges for medical devices tactile sensing and displays. In *Haptic Feedback for Minimally Invasive Surgery and Robotics*, 2012.

[21] G. Spinoglio and A. Marano. *Robotic surgery: Current applications and new trends*. Springer, 2015.

[22] M. Yip, S. Yuen, and R. Howe. A robust uniaxial force sensor for minimally invasive surgery. *IEEE Transactions on Biomedical Engineering*, 57(5):1008–1011, 2010.