DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

MEDICAL ROBOTICS FINAL PROJECT

IMPLEMENTATION OF IMPEDANCE CONTROL FOR A TELE-ECHOGRAPHIC SYSTEM

Students:
Matteo Germano - 1807599
Damiano Gasperini - 1796880
Federica Cocci - 1802435
Caterina Borzillo - 1808187

March 2022

Contents

Abstract

In our project, we were asked to implement impedance control for a manipulator with the future goal of remotely performing a commanded tele-echographic examination.

The simulated manipulator is equipped with an ultra-sound probe which will be in contact with an abdomen phantom.

Starting from an initial position above the phantom, the probe approaches the abdomen perpendicularly and then slides on it, keeping low contact forces.

1 Introduction

Echography examination offers quick and reliable non-invasive diagnosis for many pathological situations. It allows a specialist to evaluate the degree of emergency for the patient but it is a highly skilled and operator dependant technique.
Nowadays tele-medicine allows us to greatly extend the concept of health care and assistance opening, thanks to the rapid development of the Internet, new scenarios offered by disruptive technologies such as robotics overcoming also the physical limits dictated by geography and the territory in which one lives.
The first telemedical studies to perform a remote examination were reported in mid 1990s and they consisted in the transmission of ultrasound video images acquired by a technician close to the patient to a medical expert.
Telerobotic systems are classified into short-distance ones (master and slave systems located in the same room) and those operating in long physical distance. For long-distance telerobotic systems we refer to systems in which the master and slave sites are geographically separated. The main difference with short distance telerobotic system is the data link between patient site (slave robot) and the expert site (master station).
In February 2020, Professor Peng Chengzhong from Zhejiang Provincial People's Hospital used a robotic system for remote ultrasound diagnosis of a suspected case of COVID-19 pneumonia in a patient hospitalized 60 km away in the Tongxiang district.
According to an analysis conducted by Capterra in Italy, 86% of the Italians interviewed in the sample used telemedicine for the first time during COVID-19.

2 State of the Art

In this chapter, we will analyze some tele-echography's related works and system. Actually we will focus on four systems going deeper into their descriptions: to name other systems, we have also MIDSTEP, RUDS, SYRTECH, TERESA, ESTELE.
Just for curiosity, at the end of this chapter we have reported the way of how ecography is done in the space inside the International Space Station.

2.1 The TER system

Among the various tele-ultrasound systems, we introduce TER which is a tele-robotic system that is constituted by a master workstation (that can be equipped or not with a force feedback control) and by a slave robot which is suited to be tele-operated by an expert clinician who has the task of remotely performing the ultrasound based diagnosis.

Concerning the architecture and functionalities of the system, the master system has been developed and tested using 2 different solutions: the first one uses a position sensor associated with visual feedback while the second one uses an haptic device that adds a force feedback control to the system. Moreover, a peculiarity of TER is that the slave system is actuated by artificial muscles which make it naturally compliant.

A fundamental aspect that has to be taken in consideration is that, despite the fact that the current tele-operated system is still under development, it is a common thought to believe that the study and the evolution of original robotic architectures and tele-gesture capabilities are experiencing positive feedback around the world thanks to the benefits they bring in terms of quality.

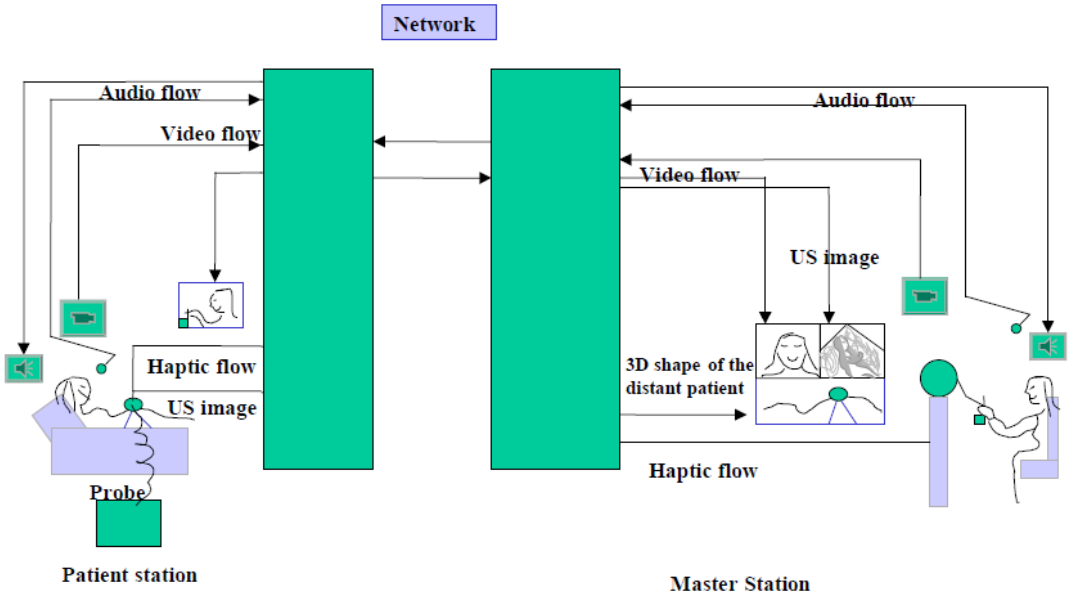


Figure 1: TER general architecture

2.2 Long-distance paradigm: the MELODY system

Another example of ultrasound robotic system is called MELODY. It follows the long-distance paradigm according to which the system has the capacity to accommodate remote, teleoperated US examination, for example from a main hospital to a local hospital or to an isolated location. Among the benefits of this mechanism there are the access to expert healthcare for remote communities and the cost-effective service delivery.

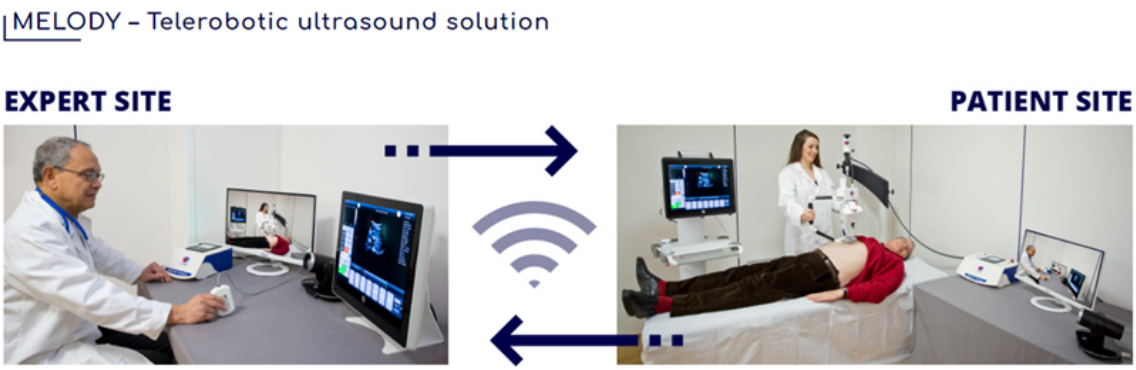


Figure 2: MELODY

The system consists of three main parts: the expert system (master station), the patient system (slave station), and the communication link that enables data exchange between the two stations. MELODY's system architecture can be seen in the image below.

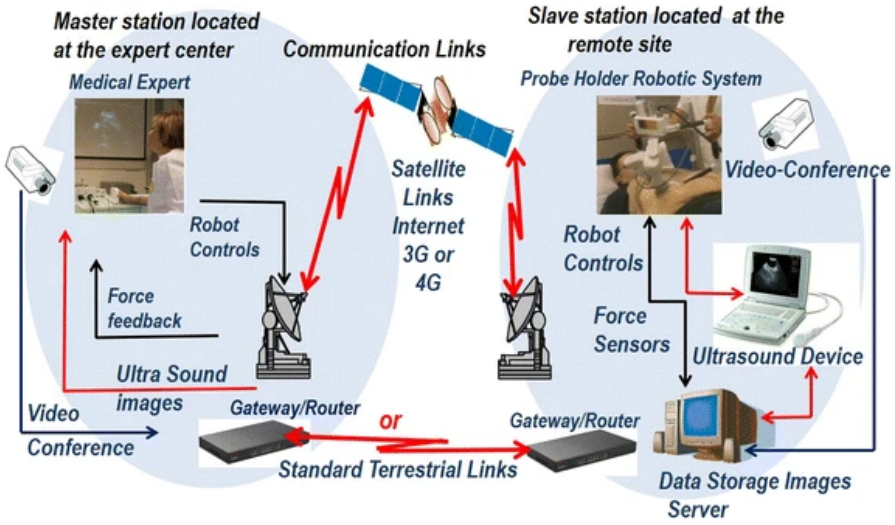


Figure 3: Robotized tele-ultrasound using the MELODY system

To conclude, it's important to point out that, despite remarkable achievements demonstrated by groups of telerobotic systems, not so many of them have been adopted in clinical practice and therefore further efforts are required to address both clinical and technological challenges.

2.3 The OTELO system

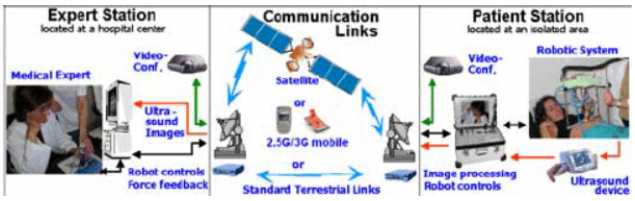
OTELO is a remotely controlled system designed to achieve reliable ultrasound imaging at an isolated site, distant from a specialist clinician. In particular it is a mobile tele-echography which

uses an ultra-light robot and it is composed of three main parts:

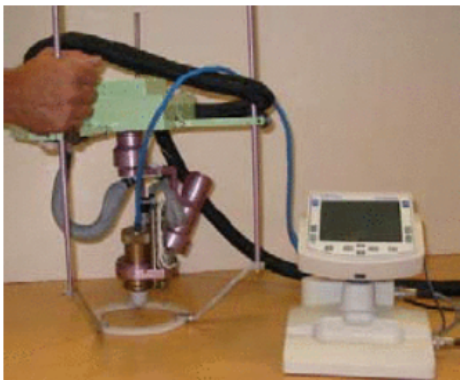
1. An "expert" site where the doctor interacts with a dedicated haptic probe to control the positioning of the remote robot. This probe emulates an ultrasound probe that medical experts are used to handle, thus providing a better ergonomy. The doctor receives also information about the contact force between the probe and the skin (a force sensor is positioned on the probe holder system).

2. The satellite or terrestrial communication links.

3. A "patient" station is made of the 6 DOF light-weight robotic system and its control unit. This robot manipulates an ultrasound probe according to orders sent by the medical expert and it is maintained by a nurse or a non-specialist assistant in echography. The paramedic positions and holds the robot on the patient's skin following the instructions given by the medical expert by videoconference. Movements of the fictive probe are reproduced by the robot on the real probe.



(a) The OTELO Mobile Robotic System



(b) OTELO Robot with the echograph Unit

Figure 4: The OTELO system

Actually there are three version of OTELO robot. Focusing on the first version, OTELO 1, a portable compact 6 DOF probe holder robot has been designed:

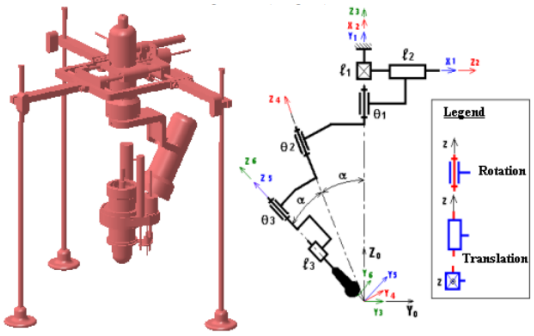


Figure 5: Mechanical structure of OTELO 1

- Three rotations of concurrent axes allow the probe to rotate inside a conical space, of 60° vertex angle, keeping a point of contact with the patient skin.
- One translation along the probe axis direction allows the probe to be continuously in contact with the skin. In addition, it allows the expert to control the contact force.
- Two translations of perpendicular axes allow the expert to adjust the position of the contact point above the organ to be investigated.

2.4 The FASTele system

The FASTele is a portable and attachable tele-echography robot system. FASTele (focused assessment with sonography for trauma) has been developed thinking about patients who have shock by internal bleeding: the patient has little time, and transportation to a hospital may take too long so develop a system which enables FAST more quickly is required. This robot system can be used by a paramedic easily for shock patient in ambulance or at injury scene. The robot is attached to each roughly FAST areas of patient body and remotely fine-tuned position by a specialist in a hospital.

The FAST robot has 4-DOF (Pitching, Rolling, Positioning and Contacting) and equips two curvature rails, soft urethane sponge, elastic silicon-based corset, two rotary motors, a linear motor and two mechanical springs. The curvature rails and rotary motors are used to achieve the pitching and the rolling of the probe. The curvature rails of 140° and 360° are used respectively to achieve 90° pitching and 360° rolling of probe. The linear motor is used to position the probe. The range that can be positioned is 100mm, the motor placed in the up layer of the pitching and rolling. Therefore, specialist can control the pitching and rolling after positioning the probe. The springs are used to generate the contact force between probe and patient body surface. In addition a corset is used to attach the robot for patient's body trunk. The robot weighs 2.2 Kg.

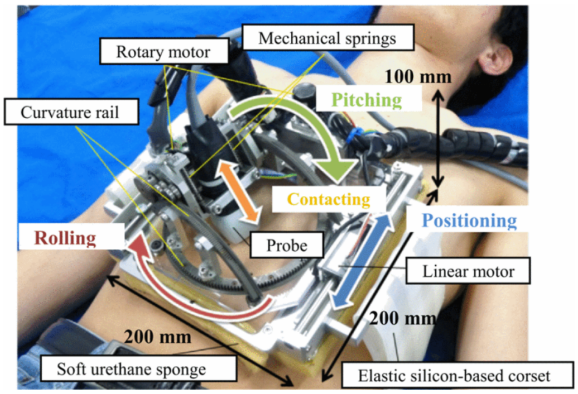


Figure 6: Attachable FAST robot

2.5 Tele-echography and space

The European Space Agency (ESA) kicked off, on 2007, the project "Advanced Robotic Tele-echography Integrated Service" (ARTIS) which was led by the Institute for Space Physiology Medicine (MEDES) in Toulouse, France. The goal of this project was to develop an end-to-end robotized tele-echography service which enables a medical expert to remotely perform ultrasound examinations on a patient from a remote site.

ARTIS pays a close attention to the usage of space infrastructures: it wants to combine different space technologies because terrestrial communications are most of the time sufficient but a satellite link is vital when examining patients located in geographically isolated areas.

At first, to evaluate this kind of telemedicine the validation has been done on patient in a sea vassel; in 2017 we have the echo inside the ISS on the astronaut Thomas Pesquet.

To allow the echography on the ISS, two tele-operated motorized probes have been developed: the first is the Abdominal Probe, the second one is the Superficial Probe. Generally speaking, at expert's site there is a basic computer connected to the Internet which is used to receive the ultrasound images and to tele-operate the two probes in the space station. The movements the expert applies on a so-called Dummy Probe are transposed on the transducer of the motorized probe at patient's site via a dedicated program. In order to do the echography in 2017, the expert guided the crew on where and how to locate the probe; as soon as the expected organ had been detected the crew was asked to keep still with the probe motionless and the probe was remotely orientated by the expert via the Dummy Probe, while the ultrasound imaging parameters were adjusted and various functions activated via a custom keyboard.

(a) Experts at the console

(b) ESA astronaut Thomas Pesquet performing the commissioning inside the ISS

(c) Expert's hand on the Dummy Probe

Figure 7: Tele-echography in space

3 Control theory

In our work we need to impose a desired dynamic behavior to the interaction between robot end-effector and the abdomen phantom. So we have decided to use cartesian impedance control. In this kind of control, the desired performance is specified through a generalized dynamic impedance, namely a complete set of mass-spring-damper equations. It is suited for tasks in which contact forces should be kept small and for this reason it is perfect for our project, consisting in the interaction with a human. Contact forces are only indirectly assigned by controlling position and the choice of a specific stiffness in the impedance model along a Cartesian direction results in a trade-off between contact forces and position accuracy in that direction. There are two kinds of impedance control: the *mechanical impedance* in which there are no force sensors that leads to a coupled behaviour (equation (6) below) and the *linear impedance* which introduce a decoupled behavior through the measurements of a force sensor. We started by considering the dynamic model of a robot in contact with a surface:

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F\dot{q} + g(q) = u - J^T(q)f_e \tag{1}$$

where f_e is the 6x1 vector of contact force and moment exerted by the manipulator's end-effector on the environment. From f_e we compute $f_A = T_A(x_e)^T f_e$ where T_A is a block matrix in this form:

$$\begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{T}(x_e) \end{bmatrix} \tag{2}$$

With reference to model equation (1), we considered the control law:

$$u = B(q)y + n(q,\dot{q}) \tag{3}$$

In the presence of end-effector forces, the controlled manipulator is described by

$$\ddot{q} = y - B^{-1}(q)J^T(q)f_e \tag{4}$$

that reveals the existence of a non linear coupling term due to the contact forces. Choosing y as

$$y = J_A^{-1}(q)M_d^{-1}(M_d\ddot{x}_d + K_D\dot{\tilde{x}} + K_p\tilde{x} - M_d\dot{J}_A(q,\dot{q})\dot{q}) \tag{5}$$

we obtain:

$$M_d\ddot{\tilde{x}} + K_D\dot{\tilde{x}} + K_p\tilde{x} = M_d B_A^{-1}(q) f_A \tag{6}$$

In the last two equations three terms appear that are J_A, B_A and $\tilde{x}$ respectively:

$$J_A = T_A^{-1} \begin{bmatrix} R^T & \mathbf{O} \\ \mathbf{O} & R^T \end{bmatrix} J(q) \tag{7}$$

$$B_A(q) = J_A^{-T}(q) B(q) J_A^{-1}(q) \tag{8}$$

and

$$\tilde{x} = x_d - x_e \tag{9}$$

4 KUKA Framework

At the very beginning we started working on a Framework implementing dynamics and kinematics of a KUKA robot. Our first goal was to install all the dependencies, compile the entire framework and get familiar with it.

4.1 Requirements

We installed the following tools:

- Visual Studio 2017 with platform toolset v140
- 64-bit V-REP software
- Fast Research Interface (FRI)
- OpenHaptics 3.5.0
- Boost 1.66.0 (boost_1_66_0-msvc-14.0-32)
- Eigen 3.4.0

Moreover we add the following environment variables to our machines:

Variables and paths	
BOOST_1_66	C:\local\boost_1_66_0
EIGEN	C:\NeedleInsertion-Framework\eigen-3.4.0\eigen-3.4.0
FRI	C:\NeedleInsertion-Framework\FRILibrary
OPEN_HAPTICS	C:\OpenHaptics\Developer\3.5.0
VREPx64	C:\Program Files\V-REP3\V-REP_PRO_EDU

4.1.1 Visual Studio and CoppeliaSim

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft; it supports 36 different programming languages.
CoppeliaSim is a robotics simulator which is based on a distributed control architecture: each object/model can be individually controlled via an embedded script, a plugin, a ROS node, a remote API client, or a custom solution. This makes CoppeliaSim very versatile and ideal for multi-robot applications. Controllers can be written in C/C++, Python, Java, Lua, Matlab or Octave: we have written using C++.

(a) Visual Studio logo

(b) CoppeliaSim logo

Figure 8: Software

4.2 Problems

The first major problem we have encountered was about a function of the framework (setJointPosition()). In fact in order to implement our control law we needed to switch to torque control while the framework was implementing a position control. So we added the function setJointTorque() as a method of VREPProxy class. In the same way as setJointPosition() and setJointTargetPosition(), this function takes as input the name of the simulated object corresponding to the joint in the VREP scene and its torque command. In particular the execution of a torque command to the joints is realized by specifying the module of the commanded value (abs(tau)) as the maximum force that a joint can realize and by specifying a high reference value in velocity or position with the same sign of the commanded torque. This reference value will be a velocity in the case that the control loop of the simulated joints is disabled, otherwise it will be a position.

The second problem faced was that in the function setJointTorque a function named simxSetJointMaxForce was used but it was not defined among the VREP ones. So we had to change from VREP to CoppeliaSim that allows the use of this required function for our project.

Once we have addressed the above problems we started to implement the impedence control law, but this one did not work for some reasons. To understand the issues we have disabled the robot probe and we have simplified the problem as a regulation task by using a dummy in the CoppeliaSim scene to have a reference position. Also in this case the movement of the robot was totally random.

Moreover we added some plots to analyze the evolution over time of commanded and desired torques. Unfortunately we noticed two completely different behaviours. For this reason we supposed there were some problems with the dynamic model of the KUKA manipulator, in particular with the dynamic parameters.

As last trial we used the easiest control law: $u = g$, to keep the manipulator in the starting position (but a drift term). Even this case failed confirming a problem with the dynamics.

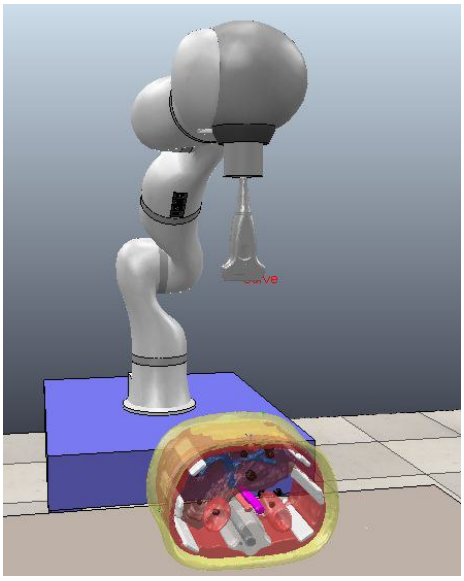


Figure 9: KUKA scene

5 FRANKA Framework

After the failure with the KUKA framework, we received the C++ files with the class of a FRANKA robot with its functionalities. From them we created a new Visual Studio project.

5.1 Project creation and dependencies

In order to use the class of the FRANKA robot we first had to compile a given Visual Studio project. The result of this compilation is a DLL library containing the dynamic model of the robot. After this step we created a new project containing all the C++ files and the corresponding headers. This new framework is simpler than the KUKA one since it is composed of a less number of files that are the only ones strictly necessary for our project. Since the DLL library was compiled with the x64 platform solution, also the entire framework must be compiled in the same way, while the KUKA one was compiled with the x86 platform solution. Regarding the dependencies, they are the same of the KUKA robot.

5.2 Problems

This time the scene in CoppeliaSim contains only the manipulator without the phantom, the end-effector/probe and the force sensor. Moreover, the force sensor class was not in the code, so we had to write it. We started with the easiest case of torque control $u = g$ and we checked that all was working. We then decided to proceed step by step by increasing the complexity of the control law. We implemented the regulation case giving the reference position of a dummy in the scene: by hardly tuning the proportional and damping gains, we succeeded in this task. We then added a table with a small cube on it in the CoppeliaSim scene to simulate a contact between the force

sensor mounted on the manipulator and the cube surface. We used a cube in a first attempt as an easy case of study because the planar surface of the cube with respect of the curved surface of the phantom is more comfortable to work with. We kept the regulation control law moving the dummy just below the cube surface but as soon as the end-effector approached the surface, the manipulator started to move in a random way. Even tuning the gains with many different values the task failed. For example we considered the block gain K as a positive definite diagonal matrix with a lower value in the third column since it corresponds to the z axis and so to the approaching direction of the end-effector to the surface in which we want low contact forces. In our opinion there could be several problems related to the CoppeliaSim scene or to the robot's dynamic model. To be even clearer we coded the regulation control law both in sensorless case and in the force sensor case and both failed. In particular we focused on only the cartesian position error, without considering orientations at that time. Obviously the use of a force sensor introduces major possible errors in the dynamic model. In fact to dynamically enable the force sensor in the CoppeliaSim scene we needed to add it a mass: we had to change the end-effector dynamic parameters of the robot but this did not work (even in the contactless case). Moreover the sensor mass must be subtracted in a proper way to the sensor measurements and these could be not precise because of a wrong dimension of the dynamic parameters of the sensor itself in the scene. Finally we noticed that in the received scene not all the links of the robot were dynamically enabled and respondable, introducing other possible errors.

For these problems we decided to change approach as described in the next section.

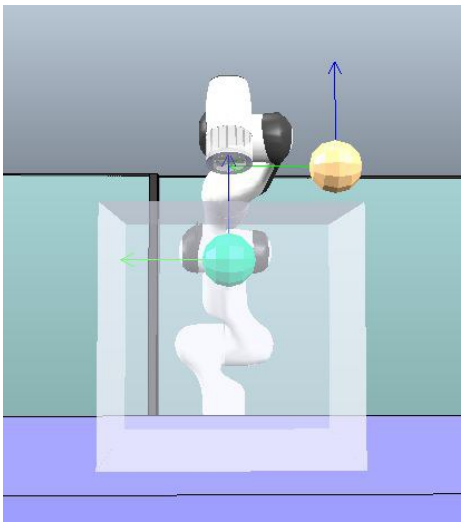


Figure 10: FRANKA scene

6 FRANKA Framework with ROS

6.1 Development

This new framework is built using catkin tools, the middleware ROS and the VISP library.

6.1.1 ViSP

ViSP library allows to control a real Panda robot from Franka Emika to perform several kinds of controllers. visp_ros is an extension of ViSP library developed by Inria Rainbow team. While ViSP is independent to ROS, in visp_ros we benefit from ROS features. It makes possible to use ROS in a transparent way, either by building classical binaries without catkin, either by building ROS nodes with catkin but without the need to write ROS specific code. Using visp_ros makes possible to simulate a Franka robot in position, velocity and force control. All the rendering and sensing is done through CoppeliaSim while the geometric and dynamic model is handled in visp_ros. Communication between visp_ros and CoppeliaSim is done using ROS. The simulation is a physical simulation with a model that has been accurately identified from a real Franka robot. [10]

6.1.2 ROS e CoppeliaSim

ROS stands for Robot Operating System and it is a software platform used for robot developing and programming. It was born in 2007 at Stanford University and it is an open-source middleware.

Figure 11: ROS logo

ROS is based on some basic concepts:

- Node: it is an instance in execution of a ROS program.
- Messages: date exchanged among nodes in order to communicate.
- Topic: it is the communication system where the message published by a node (publisher) can reach the addressed node (subscriber).
- Service: it is defined by a couple of messages (one for the request and the other for the response). A node offers a service and a client uses this service sending a request message and waiting for the response. Services are used for tasks which have a defined beginning and end.
- Action: it is different from the service because the action is an evolution and so nodes want a continuous feedback about its execution.

ROS Interface is part of the CoppeliaSim API framework. ROS Interface duplicates the C/C++ ROS API with a good fidelity. ROS is a distributed pseudo operating system allowing for easy management and communication between multiple computers connected in a network. CoppeliaSim can act as a ROS node that other nodes can communicate with via ROS services, ROS publishers and ROS subscribers. ROS Interface functionality in CoppeliaSim is enabled via a plugin: libsimExtROS.*. The plugin is loaded when CoppeliaSim is launched, but the load operation will only be successful if roscore was running at that time.

6.1.3 CoppeliaSim scene

In the scene there is a FRANKA manipulator on the floor and a table with a phantom on it. The manipulator has an echographics probe in the place of end-effector. To achieve the contact task between the end-effector and the phantom we added these two objects in two different containers. The end-effector container is set as respondable, to enable the collision detection engine in CoppeliaSim, and dynamic, in order to have a mass and some inertia values, namely the principal moments of inertia. On the other hand the phantom container is only set to be respondable because it is just a static object on which we want to observe the contact. The phantom and the end-effector have convex hull containers because they have a concave shape and CoppeliaSim can not simulate the contact between two concave objects. Actually the real contact happens between the convex hulls.

(a) Scene (b) Convex hulls

Figure 12: CoppeliaSim scene

We set a simulation step time of 3 ms to have a smooth simulation of the dynamic model, updating the dynamic parameters and matrices with a greater frequency. In our scene we decided to use the Newton dynamics engine. A script is associated to the FRANKA object in the scene, in which we initialize the handlers to manage different robot components, publishers and subscribers. We want to highlight some topics:

- /coppeliasim/franka/joint_state: measured positions, velocities and torques
- /coppeliasim/franka/g0: gravity value
- /coppeliasim/franka/flMe: position xyz, orientation xyzw
- /coppeliasim/franka/tool/inertia: mass, center of mass, tool position xyz, inertia values
- /fakeFCI/joint_state: commanded torques

The first four topics have CoppeliaSim as publisher and visp_ros as subscriber while for the last topic is the opposite.

6.2 Code

Our ROS node starts instantiating the *robot* object belonging to the *vpROSRobotFrankaCoppeliasim* class of the visp_ros package. Using its properties, the scene simulation begins after that a synchronization with CoppeliaSim is performed.
Once the interaction with the simulator is ready, the node executes a position controller to bring the manipulator in a certain desired position, above the abdomen but very close to it. At the end of this, a torque controller is used for the purpose of the project and the program enters in a while loop that will end when the user will want to stop the simulation.
When loop begins, the following variables are taken: joints' position q, joints' velocity dq, robot inertia matrix B, robot Coriolis term C, robot Friction term F, robot Jacobian fJe and measured torques tau. Consequently, the pose, velocity and acceleration errors are defined. In order to better understand their definitions, let's focus a bit on theory.
Let $O_e - x_e y_e z_e$ and $O_d - x_d y_d z_d$ denote the end-effector frame and the desired frame respectively. The corresponding homogeneous transformation matrices are:

$$M_e = \begin{bmatrix} R_e & o_e \\ \mathbf{0}^T & 1 \end{bmatrix} \qquad M_d = \begin{bmatrix} R_d & o_d \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{10}$$

The position and orientation displacement of the end-effector frame with respect to the desired frame can be expressed in terms of the homogeneous transformation matrix:

$$M_e^d = (M_d)^{-1} M_e = \begin{bmatrix} R_e^d & o_{e,d}^d \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{11}$$

where $R_e^d = R_d^T R_e$ and $o_{d,e}^d = R_d^T (o_e - o_d)$.
The new error vector in the operational space can be defined as:

$$\tilde{x} = - \begin{bmatrix} o_{d,e}^d \\ \phi_{d,e} \end{bmatrix} \tag{12}$$

where $\phi_{d,e}$ is the vector of Euler angles extracted from the rotation matrix R_e^d. The minus sign depends on the fact that, for control purposes, the error is usually defined as the difference between the desired and the measured quantities.
Computing the time derivative of $o_{d,e}^d$ and of $\phi_{d,e}$, we have:

$$\dot{o}_{d,e}^d = R_d^T (\dot{o}_e - \dot{o}_d) - S(w_d^d) R_d^T (o_e - o_d)$$
$$\dot{\phi}_{d,e} = T^{-1}(\phi_{d,e}) w_{d,e}^d = T^{-1}(\phi_{d,e}) R_d^T (w_e - w_d)$$

Given that the desired quantity R_d is constant, the vector $\dot{\tilde{x}}$ can be expressed in the form:

$$\dot{\tilde{x}} = -T_A^{-1}(\phi_{d,e}) \begin{bmatrix} R_d^T & \mathbf{O} \\ \mathbf{O} & R_d^T \end{bmatrix} \left(\begin{bmatrix} \dot{o}_e \\ w_e \end{bmatrix} - \begin{bmatrix} \dot{o}_d \\ w_d \end{bmatrix} \right) = -T_A^{-1}(\phi_{d,e}) \begin{bmatrix} R_d^T & \mathbf{O} \\ \mathbf{O} & R_d^T \end{bmatrix} \left(J(q)\dot{q} - \begin{bmatrix} \dot{o}_d \\ w_d \end{bmatrix} \right) \tag{13}$$

with

$$J_{A_d}(q, \tilde{x}) = T_A^{-1}(\phi_{d,e}) \begin{bmatrix} R_d^T & \mathbf{O} \\ \mathbf{O} & R_d^T \end{bmatrix} J(q), \tag{14}$$

i.e. the analytic Jacobian corresponding to definition (12) of error in the operational space.
Our code follows this exact formulation. Before analyzing it, in order to be dimensional clear, the following variables have been mainly used in the program with the corresponding dimensions:

```
vpColVector q( 7, 0 ), dq( 7, 0 ), tau_d( 7, 0 ), C( 7, 0 ), F( 7, 0 ), tau_d0( 7, 0 ),
       tau_cmd( 7, 0 ), x_e( 6, 0 ), dx_e( 6, 0 ), dx_ed( 6, 0 ), ddx_ed( 6, 0 );
vpMatrix fJe( 6, 7 ), Ja( 6, 7 ), dJa( 6, 7 ), Ja_old( 6, 7 ), B( 7, 7 ), I7( 7, 7 ),
       Ja_pinv_B_t( 6, 7 );
vpColVector tau( 7, 0 );
vpMatrix K( 6, 6 ), D( 6, 6 ), edVf( 6, 6 );
```

since FRANKA robot is a seven joints manipulator and our task is six dimensional, realizing cartesian position and orientation.
The $\tilde{x}$ error (x_e in the code) is defined in the following lines of code:

```
fMed[1][3] = fMed0[1][3] + ( start_trajectory ? ( 0.05 * sin( 2 * M_PI * 0.3 *
       ( time_cur - time_start_trajectory ) ) ) : 0 );
fMed[2][3] = fMed0[2][3] - ( start_trajectory ? 7.5446e-02 : 0);
x_e = (vpColVector)vpPoseVector( fMed.inverse() * robot.get_fMe() );
```

where $fMed$ has been initialized to the fMe, namely the direct kinematics corresponding to the homogeneous transformation between the robot base frame and the end-effector for the current joint position, and then the desired position o_d has been subtracted to its translational vector. As result, extracting the pose vector from the $edMe$ homogeneous matrix (third line), namely the transformation matrix between the desired frame and the end-effector one, means to follow precisely equations (11) and (12).
From the first two lines we can notice that the desired position values are only defined in the y and z-axis: going more in depth, we want the manipulator to track a sinusoidal trajectory, resulting in the sliding motion on the abdomen, along the horizontal (y) axis. On the other hand, we want the manipulator to keep contact with the phantom: this has been achieved setting a constant reference value along the vertical (z) axis, that is a vertical position just below the surface of the phantom. This means that the manipulator would like to enter the surface of the phantom, but since a contact is registered before achieving that z position, it will limit itself to keep contact.

Consecutively, the $\dot{\tilde{x}}$ is defined in the following lines of code:

```
dx_ed[1] = ( start_trajectory ? (2 * M_PI * 0.3 * 0.05 * cos( 2 * M_PI * 0.3 *
       ( time_cur - time_start_trajectory ) ) ) : 0);
dx_ed[2] = 0;
edVf.insert( fMed.getRotationMatrix().t(), 0, 0 );
edVf.insert( fMed.getRotationMatrix().t(), 3, 3 );
Ja = Ta( fMed.inverse() * robot.get_fMe() ) * edVf * fJe;
dx_e = Ta( fMed.inverse() * robot.get_fMe() ) * edVf * ( dx_ed - fJe * dq );
```

The first two lines are the derivative of the sinusoidal trajectory and of the constant z value, while the others implement the definition (13). Finally, the $\ddot{x}_d$ is just the second derivative of the sinusoidal trajectory:

```
ddx_ed[1] = ( start_trajectory ? (-2 * M_PI * 0.3 * 2 * M_PI * 0.3 * 0.05 * sin( 2 * M_PI * 0.3 *
       ( time_cur - time_start_trajectory ) ) ) : 0);
```

Now we have all that we need to implement the impedance control law:

```
tau_d = B * Ja.pseudoInverse() * ( -K * ( x_e ) + D * (dx_e)-dJa * dq + ddx_ed ) + C + F -
       ( I7 - Ja.t() * Ja_pinv_B_t ) * B * dq * 100;
```

It follows the definition stated in (3) and substituting (5) in it, but with some difference:

- there is no gravity compensation term g because it is managed inside the torque controller in visp_ros

- the FRANKA robot is redundant for our task, requiring to manage its null-space configuration while performing the main task. A pure damping behavior is proposed for the null-space configuration control, aiming to damp the null-space motion. We can call τ_r the null-space control torque:

$$\tau_r = -(I - J_A^T Ja\_pinv\_B\_t)BD_n\dot{q}; \tag{15}$$

```
tau_r = - ( I7 - Ja.t() * Ja_pinv_B_t ) * B * dq * 100;
```

where the term $-dq * 100$ allows to damp the null-space motion and in which $Ja\_pinv\_B\_t$ is defined as

```
Ja_pinv_B_t = ( Ja * B.inverseByCholesky() * Ja.t() ).inverseByCholesky() * Ja *
      B.inverseByCholesky();
```

Analysing the proportional and damping gains, we set them as two block positive definite diagonal matrices:

```
K.diag( { 50 * 50, 50 * 50, 10 * 10 , 50 * 50, 20 * 20, 20 * 20 } );
D.diag( { 2 * 50, 2 * 50, 2 * 50, 2 * 20, 2 * 20, 2 * 20 } );
```

The first three diagonal values refer to the x, y and z axes: in particular we set a low z value in the K gain, so to have desired small contact forces, while the x and y values are higher because we want a good trajectory tracking.

Computed the desired torque tau_d, we set the commanded torque tau_cmd as:

```
tau_cmd = tau_d - tau_d0 * std::exp( -mu * ( time_cur - time_start_trajectory ) );
```

in order to introduce a little delay before starting the trajectory.
Finally the *tau_cmd* is passed in input to the *setForceTorque* property of the *robot* object defined in the visp_ros package, that will perform some internal computation and then it will publish the command through the ros topic */fakeFCI/joint_state*

7 Results and conclusions

In this project we implemented a control task to perform an echographic examination. We used the impedance control because it is very suitable for our task in which we have to maintain low contact forces for the interaction with a human body. These forces can be kept low by a tuning of the proportional gain, because it has the role of modifying the stiffness of the manipulator. The robot tracks the desired sinusoidal trajectory, simulating in this way a real echography. Our code works well both for position errors and for orientation errors. In the scene during the simulation we can observe some plots for joint positions, torque measurements, pose error and norm pose error. About the plot for torque measurements, we can easily notice when the contact happens because there is some noise on them. This occurs because objects' surface is not uniform. Moreover this noise implies an upward reaction of the probe that can be mitigated with a proper tuning of the gains. Finally concerning the plots for pose error and in particular for the z-axis position error, we can observe the desired behavior along the vertical axis: at the start of the simulation the error exponentially decreases until the probe reaches the body and so the error will remain the same.

We conclude by saying that our project, after having faced some problems and done a lot of hard work, works well.

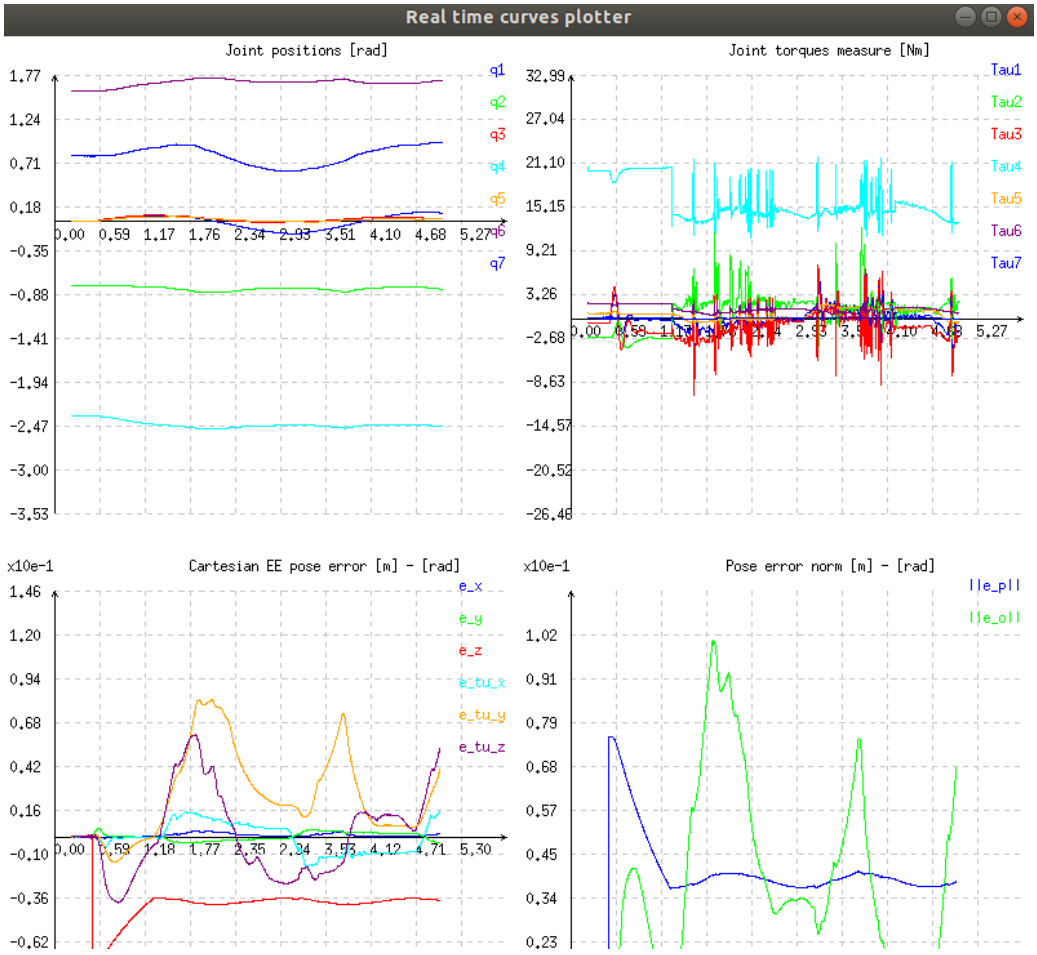


Figure 13: Plots

References

[1] A. Vilchis Gonzales, P. Cinquin, J. Troccaz1, A. Guerraz, B. Hennion, F. Pellissier, P. Thorel, F. Courreges, A. Gourdon, G. Poisson, P. Vieyres, P. Caron, O. Mérigeaux, L. Urbain, C. Daimo, S. Lavallée, P. Arbeille, M. Althuser, J. Ayoubi, B. Tondu, S. Ippolito
TER: a system for robotic tele-echography

[2] Medical telerobotic systems: current status and future trends,
`https://biomedical-engineering-online.biomedcentral.com/articles/10.1186/s12938-016-0217-7`

[3] MELODY – Telerobotic ultrasound solution,
`https://www.adechotech.com/products/`

[4] Tele-Echography Activities,
`https://www.esa.int/Enabling_Support/Preparing_for_the_Future/Space_for_Earth/Space_for_health/Tele-Echography_Activities`

[5] F. Courreges, P. Vieyres and R. S. H. Istepanian,
Advances in robotic tele-echography services - the OTELO system. The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2004.

[6] N. Smith-Guérin, L. Al Bassit, G. Poisson, C. Delgorge, Ph. Arbeille, P. Vieyres.
Clinical validation of a mobile patient-expert tele-echogrphy system using ISDN lines.

[7] K. Ito, S. Sugano and H. Iwata.
Portable and attachable tele-echography robot system: FASTele. Annual International Conference of the IEEE Engineering in Medicine and Biology, 2010.

[8] ARTIS: a step towards an end-to-end robotic tele-echography service,
`https://www.esa.int/Enabling_Support/Preparing_for_the_Future/Space_for_Earth/Space_for_health/ARTIS_a_step_towards_an_end-to-end_robotic_tele-echography_service`

[9] P. Arbeille, D. Chaput, K. Zuj, A. Depriester, A. Maillet, O. Belbis, P. Benarroche, S. Barde.
Remote Echography between a Ground Control Center and the International Space Station Using a Tele-operated Echograph with Motorized Probe. Application to isolated medical centre on the Earth. SpaceOps Conference, 2018.

[10] C. Gaz, M. Cognetti, A. Oliva, P. Robuffo Giordano, A. De Luca.
Dynamic Identification of the Franka Emika Panda Robot With Retrieval of Feasible Parameters Using Penalty-Based Optimization. IEEE RA-L, 2019.