University of Verona

DEPARTMENT OF COMPUTER SCIENCE
Master Degree in Computer Science and Engineering

Model-based design of at-the-edge heterogeneous Robotics Applications based on ROS

Candidate:
Simone Girardi

Thesis advisor:
Prof. Nicola Bombieri

Research supervisor:
Prof. Franco Fummi

Thesis submitted in 2020

Abstract

Developing distributed robotics applications on embedded devices, we have to deal with the diversity of the applications and the different platforms where these applications run. At the state of the art there are some solitutions that allow us to develop robotics applications and deploy them on embedded boards. The problem is that none of these solutions allows us to be sufficiently accurate to guarantee the functioning of the entire system, esapecially if we want to increase its complexity. To solve the problem we must to take account of the necessary reasources to run the applications and the constraints imposed by the limits of the devices.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Thesis outline

This thesis is organised in two main workflows: *Verification* and *Methodology*. Chapter 2 is an overview over the research contribution in the field of deep learning and the other tools discussed in the next chapter of this thesis. The first workflow is described in chapter 3, where a complete inspection and verification was made on the ORB SLAM algorithm to garantee a deterministic behaviour in a sequential contex. The second workflow described in chapter 4 proposes a solution to the problem of integrating heterogeneous robotic applications. With the support of edge computing platforms like Kubeedge, some experiments was made to deploy automatically our integrated system from the host (cloud) to the edge. Finally in chapter 5 the experimental results are showed and discussed.

Chapter 2

Background

2.1 Deep Learning

2.1.1 Motivations

Deep learning has recently been highly successful in machine learning across a variety of application domains, including computer vision, natural language processing, and big data analysis, among others. For example, deep learning methods have consistently outperformed traditional methods for object recognition and detection in the ISLVRC Computer Vision Competition since 2012 [26]. However, deep learning's high accuracy comes at the expense of high computational and memory requirements for both the training and inference phases of deep learning. Training a deep learning model is space and computationally expensive due to millions of parameters that need to be iteratively refined over multiple time epochs. Inference is computationally expensive due to the potentially high dimensionality of the input data (e.g., a high-resolution image) and millions of computations that need to be performed on the input data.

2.1.2 Definitions

As described in [12], the modern term "deep learning" goes beyond the neuroscientific perspective engineering applications on the current breed of machine learning models. It appeals to a more general principle of learning *multiple levels of composition*, which can be applied in machine learning frameworks that are not necessarily neurally inspired. A deep learning prediction algorithm, consists of a number of layers, as shown in Fig. 2.1.

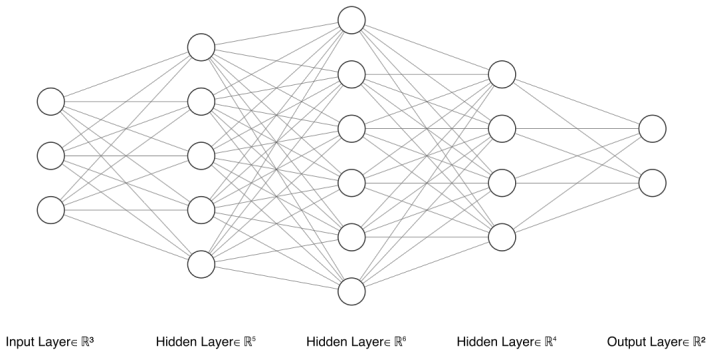


Figure 2.1: DNN example with image classification

In deep learning *inference*, the input data pass through the node's layers in sequence, and each layer performs matrix multiplications on the data. The output of a layer is usually the input to the subsequent layer. After data are processed by the final (fully connected) layer, the output is either a feature or a classification value. When the model contains many layers in sequence, the neural network is known as a deep neural network (DNN). When the matrix multiplications include convolutional filter operations, the model is named convolutional neural networks (CNNs), which is common for image and video processing contexts. There are also DNNs designed especially for time series prediction; these are called recurrent neural networks (RNNs), which have loops in their layer connections to keep state and enable predictions on sequential inputs.

In deep learning *training*, the computation proceeds in reverse order. Given the ground-truth training labels, multiple passes are made over the layers to optimize the parameters of each layer of matrix multiplications, starting from the final layer and ending with the first layer. The algorithm used is typically stochastic gradient descent (SGD). In each pass, a randomly small subset of N input data ("mini-batch") from the training data set, is selected and used to update the gradients in the direction that minimizes the training loss (where the training loss is defined as the difference between the predictions and the ground truth). One pass through the entire training

data set is called a training epoch [25].

There are some considerations to take into account: the first is that there are a large number of parameters in the matrix multiplications, resulting in many computations being performed and thus the latency issues that we see on end devices. The second is that there are many choices (hyper-parameters) on how to design the DNN models (e.g., the number of parameters per layer, and the number of layers), which makes the model design more of an art than a science. Different DNN design decisions result in tradeoffs between system metrics; for example, a DNN with higher accuracy likely requires more memory to store all the model parameters and will have higher latency because of all the matrix multiplications being performed. On the other hand, a DNN model with fewer parameters will likely execute more quickly and use less computational resources and energy, but it may not have sufficient accuracy to meet the application's requirements.

2.1.3 Performance Measurement

How can we evaluate the performance of a neural network? Deep learning can be used to perform both supervised learning and unsupervised learning. The metrics of success depend on the particular application domain where deep learning is being applied. For example, in object detection, the accuracy may be measured by the mean average precision (mAP) [26], which measures how well the predicted object location overlaps with the groundtruth location, averaged across multiple categories of objects. In machine translation, the accuracy can be measured by the bilingual evaluation understudy score metric [21], which compares a candidate translation with several groundtruth reference translations. Other general system performance metrics not specific to the application include throughput, latency, and energy. These metrics are summarized in Table 2.1. Designing a good DNN model or selecting the right DNN model for a given application is challenging due to the large number of hyperparameter decisions.

Machine learning research typically focuses on accuracy metrics, and their system performance results are often reported from powerful server testbeds equipped with GPUs. For example, Huang et al. [14] compared the speed and accuracy tradeoffs when running on a high-end gaming GPU (NVIDIA Titan X). The YOLO DNN model [24], which is designed for real-time performance, provides timing measurements on the same server

Metric	Unit
Latency	s
Energy	mW, J
Concurrent Requests Served	#
Network Bandwidth	Mbps
Accuracy	Application Specific

Table 2.1: Neural Network Perfomance Metrics

GPU. Specifically targeting mobile devices, Lu et al. [20] provided the measurements for a number of popular DNN models on mobile CPUs and GPUs (Nvidia TK1 and TX1). Ran et al.[23] further explored the accuracy-latency tradeoffs on mobile devices by measuring how reducing the dimensionality of the input size reduces the overall accuracy and latency. DNN models designed specifically for mobile devices, such as MobileNets [13], report system performance in terms of a number of multiply–add operations, which could be used to estimate latency characteristics and other metrics on different mobile hardware, based on the processing capabilities of the hardware. Once the system performance is understood, the application developer can choose the right model.

2.1.4 Frameworks

Several open-source software libraries are publicly available for deep learning inference and training on end devices and edge servers. Google's TensorFlow [4], released in 2015, is an interface for expressing machine learning algorithms and an implementation for executing such algorithms on heterogeneous distributed systems. Tensorflow's computation workflow is modeled as a directed graph and utilizes a placement algorithm to distribute computation tasks based on the estimated or measured execution time and communication time [5]. The placement algorithm uses a greedy approach that places a computation task on the node that is expected to complete the computation the soonest. Tensorflow can run on edge devices, such as Raspberry Pi and smartphones. TensorFlow Lite was proposed in the late 2017 [1], which is an optimized version of Tensorflow for mobile and embedded devices, with mobile GPU support added in early 2019. Tensorflow Lite only provides on-device inference abilities, not training, and

achieves low latency by compressing a pre-trained DNN model. Caffe [15] is another deep learning framework, originally developed by Jia, with the current version, Caffe2, maintained by Facebook. It seeks to provide an easy and straightforward way for deep learning with a focus on mobile devices, including smartphones and Raspberry Pis. PyTorch [22] is another deep learning platform developed by Facebook, with its main goal differing from Caffe2 in which it focuses on the integration of research proto- types to production development. Actually Facebook is working on the merge of Caffe2 and PyTorch frameworks. GPUs are an important key element in efficient DNN inference and training. NVIDIA provides GPU software libraries to make use of NVIDIA GPUs, such as CUDA [7] for general GPU processing and cuDNN [8] which is targeted toward deep learning. While such libraries are useful for training DNN models on a desktop server, cuDNN and CUDA are not widely available on current mobile devices such as smartphones. To utilize smartphone GPUs, Android devel- opers can currently make use of Tensorflow Lite, which provides experimental GPU capabilities. To experiment with edge devices other than smartphones, researchers can turn to edge-specific development kits, such as the Nvidia Jetson TX2 development kit for experimenting with edge computing (e.g., as used in [33]), with Nvidia-provided SDKs used to program the devices.

2.1.5 Challenges

Because of the required competences and effort to choose the best neural network and the best parameters that better fit the application of interest, there has also been much recent interest in automated machine learning, which uses artificial intelligence to choose which DNN model to run and tune the hyperparameters. For example, Tan et al. [29] and Taylor et al. [30] proposed using reinforcement learning and traditional machine learning, respectively, to choose the right hyperparameters for mobile devices, which is useful in edge scenarios.

2.2 Edge Computing

2.3 Docker

2.4 Kubernetes

Chapter 3

Verification

This chapter will focus ORB-SLAM2 description and verification. Primarly Ii will describe how ORB-SLAM2 works. Secondly it will show an accurate inspection necessary to find some potential bugs that cause an unexpected behaviour of the algorithm on a sequential enviroment.

3.1 System description

As described in [27] ORB-SLAM2 is a Simultaneus Localization And Mapping system that can works with data coming from monocular, stereo, and RGB-D cameras. A system of this kind has the purpose to allow both map reconstruction and navigation in the most common enviroment without the support of a GPS. The system consists of the following three main blocks (see 3.1):

Tracking and localization This block is in charge of computing visual features, localizing the robot in the envi- ronment, and, in case of significant discrepancies between an already saved map and the input stream, communicating updated map information to the mapping block. The frames per second (FPS) that can be computed by the whole system strongly depends on the performance of this block.

Mapping It updates the environment map by using the information (map changes) sent by the localization block. It is a computational time consuming block and its execution rate strictly depends on the agent speed. However, consider- ing the actual agent speed of the KITTI

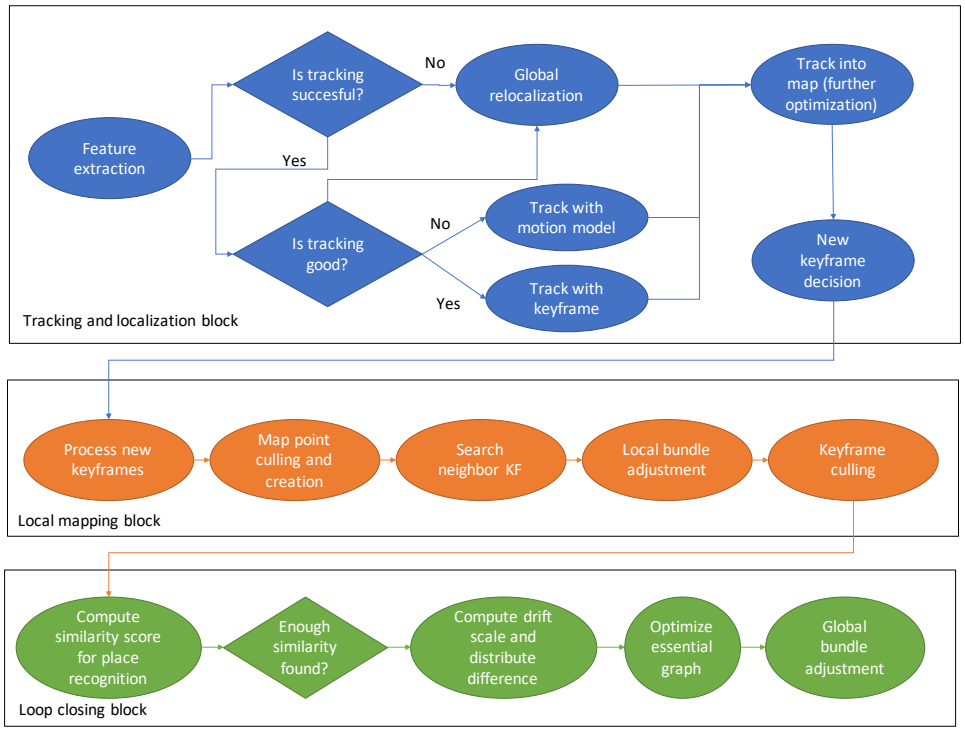


Figure 3.1: Main blocks of the ORB-SLAM2 algorithm.

datasets analysed in this work [11], it does not represent a system bottleneck.

Loop closing It aims at adjusting the scale drift error accumulated during the input analysis. When a loop in the robot pathway is detected, this block updates the mapped information through a high latency heavy computation, dur- ing which the first two blocks are suspended. This can lead the robot to loose tracking and localization information and, as a consequence, the robot to get temporary lost. The computation efficiency of this block (running on-demand) is crucial for the quality of the final results.

At each side, master and slave, a local controller is implemented that combines local and remote information to create a bilateral coupling: the operator actuates the slave while feeling a force feedback from the remote site.

The communication channel could be affected by delays: this means that a command sent from the master at time t is received by the slave at time

$t + \tau_{m2s}$, where τ_{m2s} is the amount of time required to transmit the information in the feed forward channel.
Consequently a feedback produced by the slave at time t is received at the master at time $t + \tau_{s2m}$, with τ_{s2m} the amount of time required to transmit the information in the feedback channel. The sum of τ_{m2s} and τ_{s2m} is denoted as *round-trip time* delay.
A good bilateral teleoperation system should ensure the operator a transparent and stable interface with the environment, allowing safe and immediate responses to any event both known and unknown.

From a theoretical point of view, the two main properties for a bilateral teleoperation system are:

Stability The system should maintain the stability of the closed control loop independently of operator's or environment's behaviour. Stability of the whole system is hard and complex to prove, thus this property is more easily investigate with the passivity theory.

Transparency Is the capability of the system to provide to the operator the sense of telepresence while interacting with the remote environment.

3.2 Stability

There is no single concept of stability, and many different definitions are possible. The following are fundamental statements for stability:

Definition 3.2.1. *An equilibrium state $x = 0$ is said to be:*

(a) **Stable** *if for any positive scalar ε there exists a positive scalar δ such that $\|x(t_0)\| < \delta$ implies $\|x(t)\| < \varepsilon$ for all $t \geq t_0$.*

(b) **Asymptotically stable** *if it is stable and if in addition $x(t) \to 0$ as $t \to \infty$*

(c) **Unstable** *if it is not stable.*

The definition (a) is often called "stability in the sense of Lyapunov" (stability i.s.L.) after the Russian mathematician Aleksandr M. Lyapunov (1857- 1918), whose important work features prominently in current control theory. Asymptotic stability in the large implies that all motions are bounded. Generally,

Definition 3.2.2. *An equilibrium state $x = 0$ is said to be* ***bounded*** *(or Lagrange stable) if there exists a constant M, which may depend on t_0 and $x(t_0)$, such that $\|x(t)\| \leq M$ for all $t \geq t_0$.*

To provide a proof of passivity the aim is to determine the stability nature of the equilibrium state (at the origin) of system Σ without obtaining the solution $x(\cdot)$. This could be done with the so called *direct method of Lyapunov* in relation to the (initialized) nonlinear autonomous dynamical system Σ given by

$$\dot{x} = F(x), \quad x(0) = x_0 \in \mathbb{R}^n; \quad F(0) = 0 \tag{3.1}$$

Modifications needed to deal with the (non autonomous) case

$$\dot{x} = F(t, x), \quad x(t_0) = x_0$$

are possible.

The essential idea is to generalize the concept of energy V for a conservative system in mechanics, where a well-known result states that an equilibrium point is stable if the energy is minimum.
Thus V is a positive function which has $\dot{V}$ negative in the neighbourhood of a stable equilibrium point.
More generally,

Definition 3.2.3. *We define a Lyapunov function $V : \mathbb{R}^n \to \mathbb{R}$ as follows :*

- *V and all its partial derivatives $\dfrac{\partial V}{\partial x_i}$ are continuous*
- *V is positive definite; that is, $V(0) = 0$ and $V(x) > 0$ for $x \neq 0$ in some neighbourhood $\{x \mid \|x\| \leq k\}$ of the origin*

A Lyapunov function V for the system (3.1) is said to be

- **strong** if the derivative $\dot{V}$ is *negative definite*;
  that is, $\dot{V}(0) = 0$ and $\dot{V}(x) < 0$ for $x \neq 0$ such that $\|x\| \leq k$.
- **weak** if the derivative $\dot{V}$ is *negative semi-definite*;
  that is, $\dot{V}(0) = 0$ and $\dot{V}(x) \leq 0$ $\forall x$ such that $\|x\| \leq k$.

The statements of the two theorems of Lyapunov are :

Theorem 3.2.1. Lyapunov's First Theorem
Suppose that there is a strong Lyapunov function V for system Σ. Then system Σ is asymptotically stable.

Theorem 3.2.2. Lyapunov's Second Theorem
Suppose that there is a weak Lyapunov function V for system Σ. Then system Σ is stable.

3.3 Passivity

One method derived from control theory to ensure stability of the whole teleoperation system, even in presence of delays, is by designing each component in a way that it is passive.
By the definition found in [19], a system $G(s)$ is passive if and only if a finite amount of energy can be extracted from it; or, in other words, the energy that can be extracted by a passive system is always less or at most equal to the amount inserted. Intuitively, a passive teleoperation system cannot increase the amount of energy inserted in it by the operator and/or by the environment.
To introduce the required formulation of passivity, the following definition of extended measurable functions is needed:

Definition 3.3.1. *Let $\mathcal{L}_2(\mathbb{R}^n)$ be the space of all measurable function $f : \mathbb{R} \to \mathbb{R}^n$ for which*

$$\int_{-\infty}^{+\infty} \|f(t)\|^2 \, dx < \infty \tag{3.2}$$

Definition 3.3.2. *The space $\mathcal{L}_{2e}(\mathbb{R}^n)$, $\mathcal{L}_2(\mathbb{R}^n)$ extended, is defined as the space of all measurable function $f : \mathbb{R} \to \mathbb{R}^n$ for which*

$$f_T \in \mathcal{L}_2(\mathbb{R}^n) \quad \forall T \in [-\infty, +\infty] \tag{3.3}$$

where f_T is the truncated version of f. Then the passivity definition, as in [6], is the following:

Definition 3.3.3. *If the input in the time domain is described as $u(t) \in \mathcal{L}_{2e}(\mathbb{R}^n)$ and the output $y(t) \in \mathcal{L}_{2e}(\mathbb{R}^n)$,*
a system

$$G : \mathcal{L}_{2e}(\mathbb{R}^n) \to \mathcal{L}_{2e}(\mathbb{R}^n) \quad u \to v = G(u)$$

is passive if there exist a constant $\beta > 0$ such that

$$\int_0^t y^T(\tau) u(\tau) d\tau \geq -\beta \qquad \forall t \geq 0, \forall u(\cdot) \tag{3.4}$$

The combination of two passive systems, connected in a feedback or parallel configuration, is a passive system as well. The connection of passive systems in a series configuration may not result in a passive system.
To show that passivity implies stability, we recall the definition of stability for a linear, time-invariant system $G(s)$ performed by the analysis of poles and zeroes of the system's transfer function in the Real and Imaginary plane.

Definition 3.3.4. *A linear, time-invariant system $G(s)$ is defined **stable** if and only if it does not present any pole in $\mathbb{R}^+$ plane and the poles in $\mathbb{R}^0$ are simple.*

The previous considering the imaginary axis and excluding $\mathbb{R}^+ \cup \mathbb{R}^0$ plane result in the **asymptotical stability** definition. The following theorem conveys these formulations for passivity and stability, proving that the former implies the latter:

Theorem 3.3.1. *Let $G(s)$ be the transfer function of a linear, time-invariant (LTI), single input/single output (SISO) system. G(s) is passive if and only if*

- *$G(s)$ has no pole in the $\mathbb{R}^+$ plane*
- *$ReG(j\omega) \geq 0, \quad \forall \omega \in [-\infty, +\infty]$ such that $j\omega$ is not a pole for $G(s)$*
- *if $j\omega_0$ is a pole of $G(s)$, then it must be simple and the residual be greater than 0*

$$\lim(s - j\omega_0)G(s) \geq 0$$

3.4 Teleoperation without delay

Most teleoperation algorithms has been designed to be robust even in an environment were communication delays occur. Despite that, some critical applications require absolute stability and high reliable network conditions.

3.4.1 The Four Channel Architecture

A solution, namely *The Four channel Architecture* was proposed by Lawrence [16].
It's a PF-PF (Position Force) architecture using forces and velocities as reference signals.

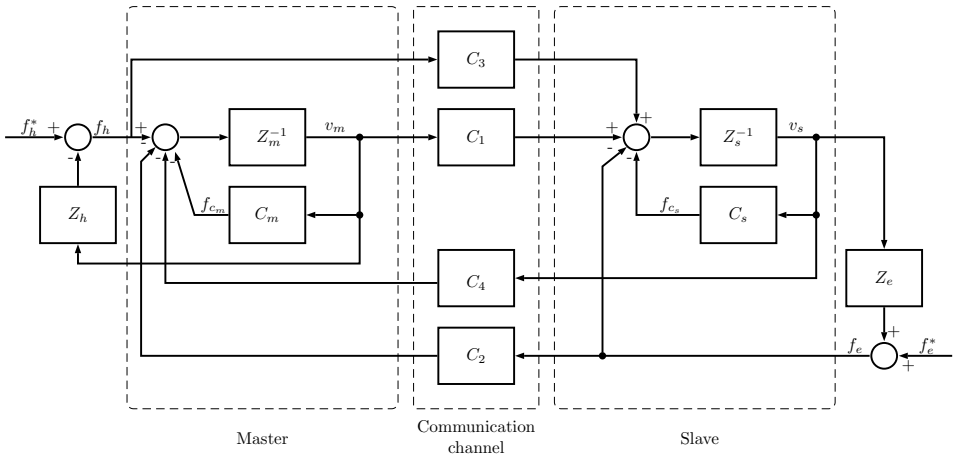


Figure 3.2: The Four Channel Architecture.

The teleoperation system is completely transparent if the operator feels that is directly interacting with the remote environment: this implies equality between forces ($F_m = F_s$) and velocities ($V_m = V_s$).
Transparency requires that the transmitted impedance Z_t is equal to the environment impedance $Z_e = F_s$. According to the block diagram in Figure3.2 the controllers $C_s, C_m, C_1, \dots C_4$ have to be designed in such a way that the hybrid matrix H

$$\begin{bmatrix} F_m(s) \\ V_m(s) \end{bmatrix} = \underbrace{\begin{bmatrix} H_{11}(s) & H_{12}(s) \\ H_{21}(s) & H_{22}(s) \end{bmatrix}}_{\triangleq H} \begin{bmatrix} V_s(s) \\ F_s(s) \end{bmatrix} \tag{3.5}$$

is equal to $\begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}$.

That implies

$$Z_t = \frac{F_m}{V_m} = (H_{11} - H_{12}Z_e)(H_{21} - H_{22}Z_e)^{-1} = Z_e \tag{3.6}$$

To achieve the goal of transparency, it is necessary to have a very accurate model both of master and slave robots. A good transparency is required mainly at low frequencies, where mathematical models are more accurate, in order to provide a good feedback at frequencies the operators work on. In his work, Lawrence proposes to associate the abstract notion of passivity derived from the mathematical representation of positive real transfer functions to a more physical measure of passivity that describes how the covariant variables at each port must be in phase in order to maintain a positive energy flow to the system.

3.5 Teleoperation with constant delay

The transmission of power variables over a communication channel affected by delays, is the reason for which bilateral teleoperation models may become unstable. Consider a basic control system structure, the mathematical description clearly shows the effect of the delays over the communication channel. Considering an input and output $X(s)$, $Y(s)$ in the Laplace domain variable s, a plant $P(s)$, a controller $C(s)$ and a feedback delay of T seconds, the transfer function of $G(s)$ is:

$$G(s) = \frac{Y(s)}{X(s)} = \frac{e^{sT}P(s)C(s)}{1 + e^{sT}P(s)C(s)} \tag{3.7}$$

The term e^{sT} that appears at denominator is the cause of the performance loss. Increasing the value of T, the system loses phase margin, thus the system could become unstable.
With a passivity based analysis the communication channel could be separated from the master and slave blocks allowing the demonstration of system's stability by passivating the channel.

3.5.1 PD and passivity terms

Lee and Spong [17] provides a solution for the teleoperation with constant time delay. It guarantees the stability of a position-position bilateral teleoperator by adding a dissipative term to the PD controller at each side

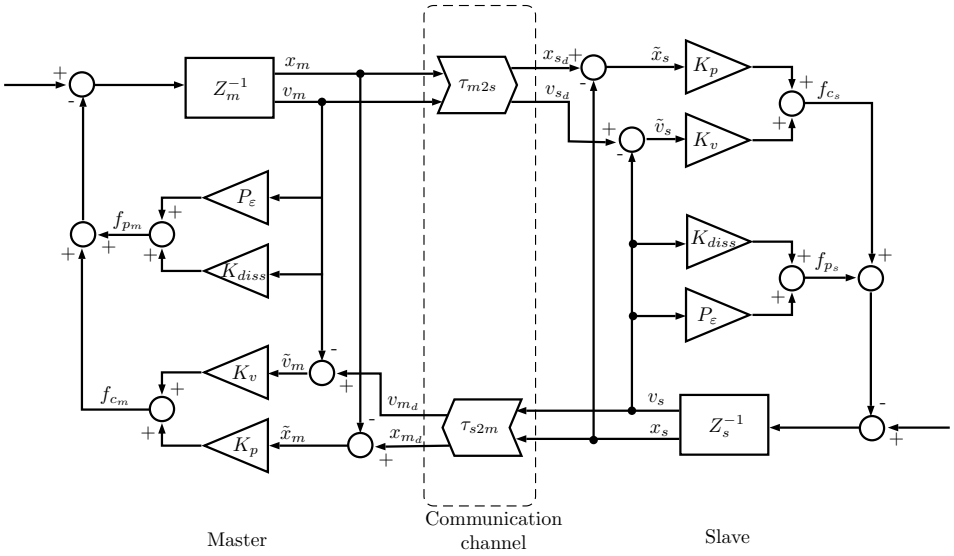


Figure 3.3: Lee-Spong PD dissipative approach

of the architecture. Figure 3.3 shows the block-schema of this architecture. This solution does not suffer of position drift between master and slave, which is the major drawback of scattering-based solutions due to the extraction of velocity from the communicated scattering variables [17].
The explicit position feedback enables to guarantee the asymptotic master-slave position coordination. This approach passifies the combination of the communication and control blocks all together, in contrast to scattering approach, where the delayed communication block is passified in such a way that the closed-loop teleoperator becomes an interconnection of passive submodules. Stability is guaranteed by an over estimation of the round-trip time delay $\bar{\tau}_{RTT} \geq \tau_{RTT}$.
Representing a manipulator with:

$$M(q)\ddot{q}(t) + C(q,\dot{q})\dot{q} = T(t) + F(t) \tag{3.8}$$

where $q \in \mathbb{R}^m$ is the configuration, $F \in \mathbb{R}^n$ is the interacting force, $T \in \mathbb{R}^n$ the control, $M(q) \in \mathbb{R}^{m\times m}$ the symmetric and positive-definite inertia matrix and $C(q,\dot{q}) \in \mathbb{R}^{m\times m}$, the control terms for master and slave could be written as function of local and remote delayed informations as follow:

$$T_m(t) := T_m\left(q_m(t),\ \dot{q}_m(t),\ q_s(t-\tau_s),\ \dot{q}_s(t-\tau_s)\right) \in \mathbb{R}^m \tag{3.9}$$

$$T_s(t) := T_s\left(q_s(t),\ \dot{q}_s(t),\ q_m(t-\tau_m),\ \dot{q}_m(t-\tau_{\mathsf{m}})\right) \in \mathbb{R}^m \tag{3.10}$$

The goal is to design the control terms T_s and T_m to achieve

- **master-slave position coordination** if $(F_m, F_s) = 0$, then

$$q_{error}(t) := q_m(t) - q_s(t) \to 0, \quad t \to 0 \tag{3.11}$$

- **static force reflection** if $(\dot{q}_m(t),\ \dot{q}_s(t),\ q_m(t),\ q_s(t)) \to 0$, then

$$F_m(t) \to -F_s(t) \tag{3.12}$$

- **energetic passivity** of the closed loop teleoperation system. It there exists a finite constant $d \in \mathbb{R}^m$ such that

$$\int_0^t \left[F_m^T(\theta)\dot{q}_m(\theta) + F_s^T(\theta)\dot{q}_s(\theta)\right] d\theta \geq d^2 \quad \forall t \geq 0 \tag{3.13}$$

  meaning that the maximum extractable energy from the two-port closed-loop teleoperator is always bounded.

In order to fulfil (3.11), (3.12) and (3.13) the design of master and slave controls $T_m(t)$ and $T_s(t)$ in (3.9) and (3.10) is:

$$T_m(t) = \underbrace{\underbrace{-K_p((q_m(t) - q_s(t-\tau_m))}_{\text{delayed P-action}}\underbrace{-K_v(\dot{q}_m(t) - \dot{q}_s(t-\tau_m)}_{\text{delayed D-action}}}_{F_{mc}} \underbrace{\underbrace{-\left(K_d + P_\varepsilon\right)\left(\dot{q}_m(t)\right)}_{\text{dissipation}}}_{F_{mp}} \tag{3.14}$$

$$T_s(t) = \underbrace{\underbrace{-K_p((q_s(t) - q_m(t-\tau_s))}_{\text{delayed P-action}}\underbrace{-K_v(\dot{q}_s(t) - \dot{q}_m(t-\tau_s)}_{\text{delayed D-action}}}_{F_{sc}} \underbrace{\underbrace{-\left(K_d + P_\varepsilon\right)\left(\dot{q}_s(t)\right)}_{\text{dissipation}}}_{F_{sp}} \tag{3.15}$$

where $K_d = \dfrac{\bar{\tau}_{RTT}}{2} K_p$ and P_ε is an additional damping ensuring master-slave coordination.

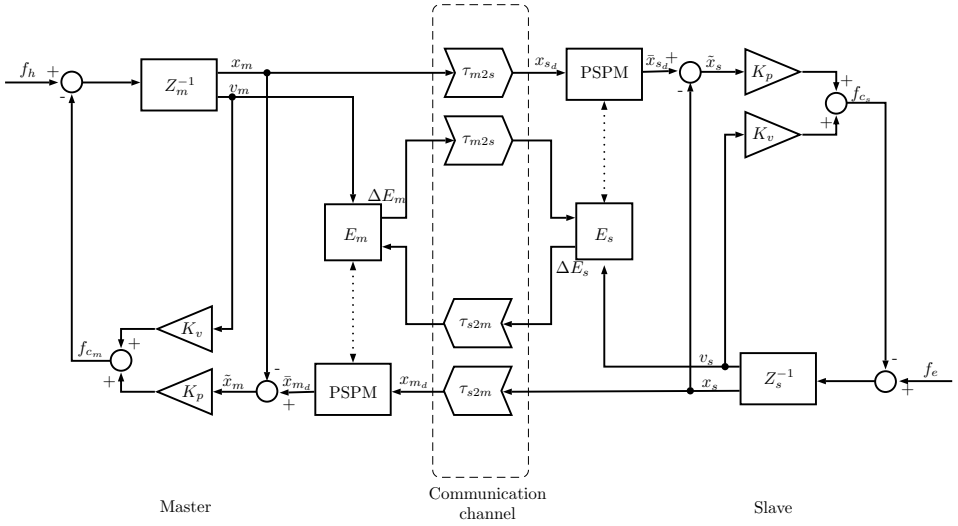


Figure 3.4: PSPM teleoperation schema

3.6 Teleoperation with time varying delay

The occurrence of time-varying delays makes the energy handling in the communications channel a challenging problem. Even the most stable and transparent architecture is susceptible to that.
The simpler solution is the use of buffers at each side to re-order information and absorb the variation by releasing packets with a constant rate. The drawbacks of this approach are the increased RTT because of the time spent for store each packet in the buffer and the dynamic growing of the buffer if the delay increases.

3.6.1 PSPM

The Passive Set-Position Modulation (PSPM) is P-P teleoperation architecture proposed by Lee [18] that addresses teleoperation with time-variant delay.

A representation is available in Figure 3.4 The input signal $y(k)$ is a discrete-time input sequence of sparse or slowly updating set position, possibly non uniformly received at time t_k.
This signal is applied to a spring-damper model $K_P\left(x(t) - y(k)\right) + K_d v(t)$ where $x(t)$ and $v(t)$ are the continuous time values of manipulator position and velocity. The approach consider the possible passivity breaking due to

the spring's energy jump at switching instances by solving the minimisation problem

$$\|x(t) - y(k)\| \quad \text{s.to} \quad E(k+1) + \Delta E_{in}(k) + D(k-1) - \Delta \bar{P}(k) \geq 0 \tag{3.16}$$

into the PSPM block each time the reference signal is updated. $E(k+1)$ is the local energy reserve , $\Delta E_{in}(k)$ is the energy received at time t_k from the counterpart (*energy-shuffling*), $D(k-1)$ is the causal approximation of the damping dissipation (*energy re-harvesting*), namely $\frac{1}{2}K_d v^2(t)$, that is position-only dependent to avoid the sequence of numerical differentiation and integration that will introduce noise contamination into the signal. Finally $-\Delta \bar{P}(k)$ is the spring energy jump

$$-\Delta \bar{P}(k) = \frac{1}{2}\|x(t_k) - \bar{y}(k)\|^2_{K_p} - \frac{1}{2}\|x(t_k) - \bar{y}(k-1)\|^2_{K_p} \tag{3.17}$$

The resulting $\bar{y}(k)$ signal is modulated in such a way that it is as close as possible to the original $y(k)$ yet only to the extent that the use of $\bar{y}(k)$ for the spring coupling K_p is permissible by the passivity constraint.
The *virtual energy reservoir* is a mechanism to improve performance by the use of *energy reharvesting*. It recaptures and deposits in the *energy reservoir* a portion of the otherwise wasted energy due to the damper and through the *energy shuffling/ceiling*. The latter limits the accumulation of virtual energy by setting an upper shorthold to the reservoir and sending the exceeding energy to the counterpart.

This approach should accommodate a variety of communication/data-update imperfections of $y(k)$, including variable-rate update, varying delay, packet loss, and even time-swapping.

3.6.2 The Two Layer Algorithm

This algorithm proposed by Franken *et al.* [10] implements a hierarchical two-layer approach: a top *transparency layer* and a bottom *passivity layer*. The main idea is that, without making any assumption about on type of controller implemented, in the former layer a control algorithm is in charge of displaying the desired behaviour and achieving transparency. The only requirement is that the algorithm implemented in this layer computes the control forces τ_{TL} to be applied to the manipulator.
The passivity layer on the bottom monitors and enforces the energy balance

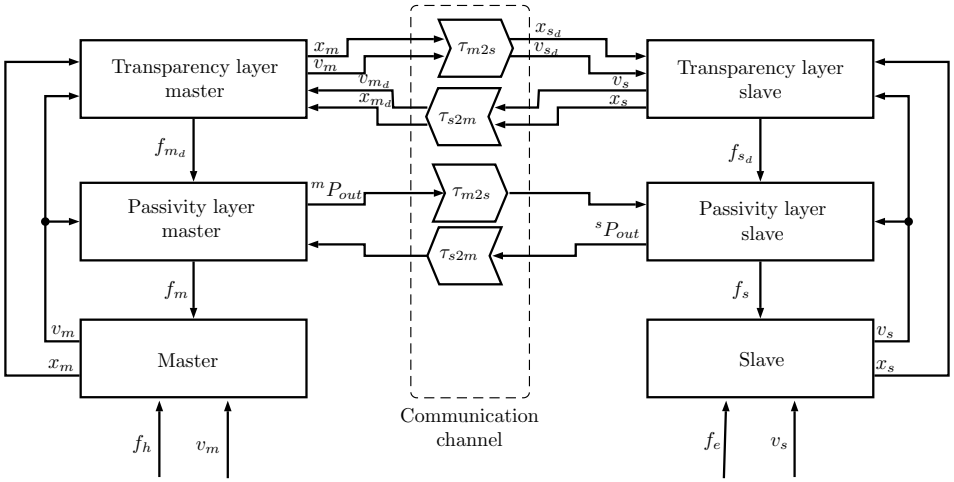


Figure 3.5: TwoLayer teleoperation schema. F_{m_d}, F_{s_d} are master and slave τ_{TL}, while f_m, f_s are the two τ_{PL}

in the system ensuring that no virtual energy is generated. With this approach, the strategy used to obtain transparency is independent of the one to obtain passivity, thus enabling the possibility to adopt in the transparency layer strategies known to be non passive e.g. most filtering techniques.
Two two-way communication channels enable the communication between master and slave: one is dedicated to communicate energy exchange information between the passivity layers, the other relays the information required by the algorithm in the transparency layer. In contrast with other approaches, the passivity is achieved after the computation of the commands and not before. This architecture is summarized in Figure3.5.

The passivity layer requires impedance causality both for master and slave (i.e. velocities as input and forces as output).
The latter layer acts as follows: at first, the energy exchange between the discrete time controller and physical world is evaluated as:

$$\begin{aligned}\Delta H(k) &= \int_{(k-1)\Delta T_s}^{(k)\Delta T_s} \tau_r(k)\dot{q}(t)dt \\ &= \tau_r(k)(q(k) - q(k-1)) \\ &= \tau_r(k)\Delta q(k)\end{aligned} \tag{3.18}$$

where $\tau_r(k)$ is the torque exerted by the actuators at sample k, $\dot{q}(k)$ the velocity vector of the actuators and $q(k)$ their sampled position.
At the same time the energy received from the counterpart is taken into

account:

$$H_+(k) = \sum_{i \in Q(k)} H(i) \tag{3.19}$$

where Q is the queue which stores all the packets received between two time instants and $H(i)$ is the i_{th} packet in the queue. Then the energy level in the local tank is updated:

$$H(k) = H(\bar{k}) + H_+(k) - \Delta H(k) \tag{3.20}$$

where $H(\bar{k})$ is the state of the tank at the previous sampling time.
An energy quantum $H_-(k) < H(k)$ could be transferred to the tank on the other side of the teleoperation system according to the transport protocol that has been implemented. $H_-(k)$ is then extracted from the tank. The energy available during the next sampling period $H(\overline{k+1})$ becomes

$$H(\overline{k+1}) = H(k) - H_-(k) \tag{3.21}$$

The available energy is then used to ensure the stability of the system.
An approach could be to allow the command τ_{TL} only if the tank stores enough energy

$$\tau_{max1}(k) = \begin{cases} 0, & \text{if } H(\overline{k+1}) \leq 0 \\ \tau_{TL}, & \text{otherwise} \end{cases}$$

or to compute an upper bound for the command in order to guarantee the stability of the teleoperated system

$$\tau_{max2}(k) = \frac{H(\overline{k+1})}{\dot{q}(\bar{k})\Delta T s} \cdot$$

The combination of multiple approaches allows more advanced passivation strategy by allowing e.g. the minimum value from a set of conditions

$$\tau_{max}(k) = \min(\tau_{max1}(k), \tau_{max2}(k), \dots)$$

Then the torques τ_{PL} are a bounded version of the ones requested by the transparency layer

$$\tau_{TL}(k) = sgn(\tau_{TL}) \min(\tau_{PL}, \tau_{max}(k)) \cdot \tag{3.22}$$

To maintain the energy level above a minimum threshold, usually at master side, a Tank Level Controller (TLC) is implemented. When the energy goes

below the threshold H_D, an extra variable damping is injected to extract an extra amount of energy

$$\tau_{TLC}(k) = -d(k)\dot{(q)}(k) \tag{3.23}$$

with $d(k)$

$$d(k) = \begin{cases} \alpha(H_D - H(\overline{k+1})), & \text{if } H(\overline{k+1}) < H_D \\ 0, & \text{otherwise} \end{cases} \tag{3.24}$$

Thus the command to be applied to the device during the sample period $k+1$ becomes

$$\tau_r(k+1) = \tau_{TL}(k) + \tau_{TLC}(k) \tag{3.25}$$

Looking at the communication channel, the change of energy in the channel $\Delta H_C(k)$ at sample k is

$$\begin{aligned} H_C(k) &= \sum_{i=1}^{k} \Delta H_C(k) \\ &= \sum_{i=1}^{k} H_{-M}(i) - H_{+M}(i) + H_{-S}(i) + H_{+S}(i) \end{aligned} \tag{3.26}$$

and due to time delays and packet loss in the channel

$$\begin{aligned} H_{-S}(i) &= H_{+M}(i + d_{SM}(i)) \\ H_{-M}(i) &= H_{+S}(i + d_{MS}(i)) \end{aligned} \tag{3.27}$$

where $d_{MS}(i) \geq 0$ and $d_{SM}(i) \geq 0$ takes into account nondeterministic time delays. Therefore

$$\begin{aligned} \sum_{i=0}^{k} H_{-M}(i) &\geq \sum_{i=0}^{k} H_{+S}(i) \\ \sum_{i=0}^{k} H_{-M}(i) &\geq \sum_{i=0}^{k} H_{+S}(i) \end{aligned} \tag{3.28}$$

so that

$$H_C(k) \geq 0 \qquad \forall k \tag{3.29}$$

meaning that the communication channel can never produce energy as long as packets duplication is prevented.

We can now write the total energy $H_T(k)$ in the control system at instant k as

$$H_T(k) = H_M(k) + H_C(k) + H_S(k) \cdot \tag{3.30}$$

Finally the passivity condition for the system is

$$H_T(k) \geq 0 \tag{3.31}$$

and the condition that ensure a passive interconnection of the entire system to the physical world is

$$\dot{H}_T(k) \leq P_M(k) + P_S(k) \tag{3.32}$$

where $\dot{H}_T(k)$ is the rate of the energy balance of the system, $P_M(k)$ and $P_S(k)$ are respectively the power of master and slave flowing between the robot and it's respective controller.

3.7 Discussion

In this chapter the problem of teleoperation has been addressed and some of the possible solutions to overcome the problem of delay in the communication and keep the system passive have been presented. Teleoperation have two main goals: stability and transparency. The former is mandatory and ensured by the passivity of the system, the latter is responsible for a good haptic feedback and for the precision while performing a task.
The strategy adopted to achieve the goal of passivity could reduce the level of transparency in the system. A flexible and robust approach between the one presented is the Two-Layer algorithm.

Chapter 4

Methodology

This chapter will present the improvements of the Two-Layer teleoperation architecture developed within the i-Sur European project [9] for a needle insertion task. First we present the teleoperation system modelled as a port-Hamiltonian system [9], then we demonstrate both the equivalence between energy computation in task and joint space, and that a proper energy scaling between master and slave keeps the whole teleoperation system passive. Finally we describe the two teleoperation schemas, a Position-Position and a Position-Force architecture. Both control architectures are endowed with *pilot mode* for insertion assistance.

4.1 A model for the teleoperation system

The port Hamiltonian framework is a generalization of standard Hamiltonian mechanics, where energetic characteristics and power exchange between subsystems are clearly identified.

4.1.1 The port-Hamiltonian description

The most common representation of a port-Hamiltonian system is

$$\begin{cases} \dot{x} &= [J(x) - R(x)] \dfrac{\partial H}{\partial x} + g(x)u \\ y &= g^T(x) \dfrac{\partial H}{\partial x} \end{cases} \tag{4.1}$$

where $x \in \mathbb{R}^n$ is the state vector and $H(x) : \mathbb{R}^n \to \mathbb{R}$ is the lower bounded Hamiltonian function representing the amount of energy stored in the system, $J(x) = -J(x)^T$ and $R(x) \geq 0$ are the matrices that represent the inter-

nal energetic interconnections and the dissipation of the port-Hamiltonian system, and $g(x)$ is the input matrix. The input u and output y are dual variables and their product is (generalized) power. The pair (u^T, y) represents the power exchanged by the system with the external world. It can be shown that the following equality holds:

$$\dot{H}(x) + \frac{\partial^T H}{\partial x} R(x) \frac{\partial H}{\partial x} = u^T(t) y(t) \tag{4.2}$$

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

$$D(x) = \frac{\partial^T H}{\partial x} R(x) \frac{\partial H}{\partial x} \geq 0 \tag{4.3}$$

be the power dissipated by the system. $D(x)$ represent the *passivity margin*: the larger $D(x)$, the higher the passivity of the system. A large passivity margin allows the system to absorb more energy generated by non passive actions while preserving its passivity.

4.1.2 Port-Hamiltonian system with energy tank

The Two-Layer framework exploits the passivity margin by using of energy tank framework. The energy dissipated by the system is stored in a (virtual) energy tank and can be used for implementing any desired control action in a passivity preserving way as described in Section 3.6.2.
The model for a port-Hamiltonian system endowed with a tank is given by

$$\begin{cases} \dot{x} &= [J(x) - R(x)] \dfrac{\partial H}{\partial x} + g(x)u \\ \dot{x}_t &= \dfrac{\sigma}{x_t} D(x) + \dfrac{1}{x_t}(\sigma P_{in} - P_{out}) + u_t \\ y_1 &= \begin{pmatrix} y \\ y_t \end{pmatrix} \end{cases} \tag{4.4}$$

where x_t is the state associated with the energy storing tank, and

$$T(x_t) = \frac{1}{2} x_t^2 \tag{4.5}$$

is the energy stored in the tank. $P_{in} \geq 0$ and $P_{out} \geq 0$ are the incoming and outgoing power flows that the tank can exchange with other tanks. The

pair (u_t, y_t) is the power port the tank can use to exchange energy with the external world and $y_t = \frac{\partial T}{\partial x_t} = x_t$. The parameter $\sigma \in \{0, 1\}$ allows to upper bound the amount of energy stored in the tank.

The power balance for the tank is derived from the (4.4):

$$\dot{T} = \sigma D(x) + \sigma P_{in} - P_{out} + u_t^T y_t \tag{4.6}$$

which means that, if $\sigma = 1$, the tank stores the power dissipated by the system $D(x)$ and the incoming power flow P_{in}, while the outgoing power flow P_{out} is released. Furthermore the power can be injected and extracted from the tank via the power port (u_t, y_t).
In order to avoid singularities,($x_t = 0$ in (4.4)) a small amount of energy must always be present in the tank, thus we chose an arbitrary small threshold $\varepsilon > 0$ representing the minimum amount of energy we want stored into it. The tank has to be initialize and managed in such a way that $T(x_0) > \varepsilon$ and energy extraction is not allowed if $T(x_0) \leq \varepsilon$. It is also necessary to set the upper bound σ mentioned before. In fact, if there is no bound, the energy available can become very large as time increases and it would be possible to implement behaviour that are unstable in practice, even if the system remain passive for a while. Thus, σ is set with the following policy:

$$\sigma = \begin{cases} 1, & \text{if } T(x_t) \leq \bar{T} \\ 0, & \text{otherwise} \end{cases} \tag{4.7}$$

where $\bar{T}$ is an application dependent upper bound on the energy that can be stored in the tank.

The energy stored in the tank can be exploited for passively implementing any desired input $w \in \mathbb{R}^n$ to the port-Hamiltonian system with the tank is associated, by connecting the power port of the system (u, y) with the power port of the tank (u_t, y_t) through the following preserving interconnection:

$$\begin{cases} u &= \dfrac{w}{x_t} y_t = \dfrac{w}{x_t} x_t = w \\ u_t &= \dfrac{-w^t}{x_t} y \end{cases} \tag{4.8}$$

that implies the balance

$$u^T y = -u_t^T y_t \tag{4.9}$$

meaning that the energy supplied to/extracted from the port-Hamiltonian system for implementing the desired input is the same extracted from/sup-

plied to the tank, consequently no energy is generated in the whole interconnected system.

4.1.3 A port-Hamiltonian formulation for the tank-based teleoperation system

The master and slave manipulators in a teleoperation system not affected by communication delays can be represented with the formalism with $i = m, s$

$$\begin{cases} \begin{pmatrix} \dot{x}_i \\ \dot{p}_i \end{pmatrix} & = \begin{pmatrix} 0 & I \\ -I & R_i \end{pmatrix} \begin{pmatrix} \dfrac{\partial H_i}{\partial x_i} \\ \dfrac{\partial H_i}{\partial p_i} \end{pmatrix} + \begin{pmatrix} 0 \\ I \end{pmatrix} F_{ext,i} + \begin{pmatrix} 0 \\ I \end{pmatrix} F_i \\ \dot{x}_{t_i} & = \dfrac{\sigma_i}{x_{t_i}} D_i(x) + \dfrac{1}{x_{t_i}} (\sigma^i P_{in} - {}^i P_{out}) + u_{t_i} \qquad i = m, s \\ y_i & = \begin{pmatrix} v_i \\ y_{t_i} \end{pmatrix} \end{cases} \tag{4.10}$$

where $x_{t_i}, y_{t_i} = x_{t_i}$, and $T_i = \frac{1}{2} x_{t_i}^2$ are the state of the tank, the output associated to the tank, and the energy stored in the tank respectively, D_i represents the energy dissipated by the robot possibly increased by inserting a local damping injection (e.g. TLC) and ${}^i P_{in}$ and ${}^i P_{out}$ are the power flows that can be exchanged with the tank. $F_{ext,i}$ are the external forces acting on the system and F_i are the command provided by the passivity layer.
The coupling forces $F_{d,i}$ provided by the transparency layer are implemented using the energy available in the tanks by the following power preserving interconnection between the tank power port (u_{t_i}, y_{t_i}) and the robot power port (F_i, v_i):

$$\begin{cases} F_i & = \dfrac{F_{d,i}}{x_{t_i}} y_{t_i} = \dfrac{F_{d,i}}{x_{t_i}} x_{t_i} = F_{d,i} \\ u_{t_i} & = -\dfrac{F_{d,i}^T}{x_{t_i}} v_i \end{cases} \qquad i = m, s \tag{4.11}$$

With this interconnection the command $F_{d,i}$ provided by the transparency layer is implemented by exchanging energy with the tank: if $F_{d,i} > 0$, it means that some amount of energy is necessary for implementing the desired input ans should be extracted from the tank. If $F_{d,i} < 0$, it means that the desired action is dissipative, thus the dissipated energy is stored in the tank.

The provided input could be passively achieved if and only if the tank is full enough, otherwise it could be either modulated by some passivation strategy or completely cut off as explained in Section 3.6.2.

If the tank goes below the threshold $T(x_{t_i}) \leq \varepsilon_i$, energy extraction is forbidden and the coupling force implemented is $F_{d,i} = 0$. This ensure the passivity and a stable behaviour of the teleoperation system but, it negatively affects its transparency.
It is then necessary to ensure that, both at the master and slave side, the level of the tank is kept much higher than a more conservative minimum level. There are two methods for increasing the local tank level: transfer energy from the remote tank and extract it from the local robot by damping injection (TLC). The former should be preferred over the latter as it does not affect the dynamic of the system and does not affect either the feedback perceived by the user, as well.
The energy transfer strategy acts as follows: when the energy is below the more conservative minimum threshold ${}^iT_{req}$, an *energy quantum* is requested from the other tank. If at the other side the fill level of the tank is high enough, namely it is greater than a threshold ${}^iT_{ava}$, an *energy quantum* is sent. This *availability* threshold helps to keep the energy balanced through the two sides. Furthermore, if the tank is already full, all the energy dissipated by the local robot is sent to the other side in order to recover it. Formally the request signal is:

$$ {}^iE_{req} = \begin{cases} 1, & \text{if } T_i(x_{t_i}) < {}^iT_{req} \\ 0, & \text{otherwise} \end{cases} \qquad i = m, s \tag{4.12} $$

and the whole energy transfer strategy could be described as

$$ \begin{cases} {}^mP_{out} = (1 - \sigma_m)D_m + {}^sE_{req}\beta_m\bar{P}_m = {}^sP_{in} \\ {}^sP_{out} = (1 - \sigma_s)D_s + {}^mE_{req}\beta_s\bar{P}_s = {}^mP_{in} \end{cases} \tag{4.13} $$

where, as discussed in Section 4.1.2, if the tank is already full we send to the other tank the energy dissipated by the robot setting $\sigma_i = 0$.
The variable $\beta_i \in \{0, 1\}$ enables/disables the transfer of the energy from the tank, and its value is given by

$$ \beta_i = \begin{cases} 1, & \text{if } T_i(x_{t_i}) \geq {}^iT_{ava} \\ 0, & \text{otherwise} \cdot \end{cases} \tag{4.14} $$

$\bar{P}_i > 0$ is the rate of energy flowing from one tank to the other and it's a design parameter. The bigger $\bar{P}_i$, the faster the energy transfer.
All these thresholds are also application dependent, and the following constrain must be satisfied: $\varepsilon_i < {}^iT_{req} < {}^iT_{ava} < \bar{T}_i$, for $i = m, s$

This teleoperation schema is passive with respect to the environment, i.e. it is passive with respect to the pair $\left(\left(F_h^T, F_e^T\right)^T, \left(v_m^T, v_s^T\right)^T\right)$

Proof. [9] Consider the total energy of the teleoperation system as a storage function:

$$H(t) = H_m(t) + H_s(t) + T_m(t) + T_s(t) \tag{4.15}$$

Using (4.10) we obtain that

$$\begin{aligned} \dot{H}(t) = &- D_m(t) + F_m^T v_m + \sigma_m D_m(t) + \sigma_m {}^mP_{in} - {}^mP_{out} \\ &- D_s(t) + F_s^T v_s + \sigma_s D_s(t) + \sigma_s {}^sP_{in} - {}^sP_{out} \\ &+ u_{t_m} y_{t_m} + F_h^T v_m + u_{t_s} y_{t_s} + F_e^T v_s \end{aligned} \tag{4.16}$$

Since the tanks and the robots are interconnected with the power preserving interconnection (4.11), we have that $F_i^T v_i = -u_{t_i} y_{t_i}$, where $i = m, s$ thus the (4.16) can be rewritten as

$$\begin{aligned} \dot{H}(t) = &- (1-\sigma_m) D_m(t) - (1-\sigma_s) D_s(t) \\ &- (1-\sigma_m) {}^mP_{in} - (1-\sigma_s) {}^sP_{in} \\ &+ F_h^T v_m + F_e^T v_s \end{aligned} \tag{4.17}$$

Finally using (4.13), we have that

$$\begin{aligned} \dot{H}(t) = &- (1-\sigma_m) D_m(t) - (1-\sigma_s) D_s(t) \\ &- (1-\sigma_m) {}^sP_{out} - (1-\sigma_s) {}^mP_{out} \\ &+ F_h^T v_m + F_e^T v_s \end{aligned} \tag{4.18}$$

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

$$\dot{H}(t) \leq F_h^T v_m + F_e^T v_s \tag{4.19}$$

thus (3.32) holds, which proves our statement. □

Augmenting the local damping, via the TLC approach, is a passivity preserving technique. For a formal proof the port-Hamiltonian model should be rewritten with a variable damping matrix and the previous proof is similar.

4.2 Improvements on the Two-Layer

The Two Layer approach is a really versatile and powerful tool for the development of a passive bilateral teleoperation system. Although there is some space to exploit it, giving even more flexibility and control to the designer in order to better manage transparency, thus preserving passivity, even neglecting the dynamic model of the two manipulators involved.

4.2.1 Energy evaluation in task space

Both in the i-Sur project and in the original paper from Frankel *et. all* [10], energy computation and the passification strategy are performed in the joint space. For a single DoF device it is less challenging to develop a passification strategy and analyse how the transparency of the system is affected. But for a multi DoF device the passification strategy could heavily affect transparency: a uniform action between the joints could, due to the kinematics mapping, results in a dangerous behaviour in the task space.
Take needle insertion as an example: for ensuring the safety of the patient, when no needle steering techniques are involved, the path executed must follow the needle's main z axis. It is desirable that also the passivity strategy could, in a way, act preserving, as much as possible, transparency in the task space to ensure the patient safety. In this example could be a good idea splitting the task into two subtasks with different degrees of transparency priority.
An high degree of priority task should ensure the correct $x - y$ position and orientation of the needle tip, the other one, which could afford more safely a lower degree of transparency, performs the puncturing action along the z axis. In fact if there is not enough energy for performing the full action, the system will puncture with less force but the needle will not injure neighbour tissues.
This approach requires only minimal changes into the passivity layer: in fact the notion of transparency affects only the strategy on how τ_{max} is computed in (3.22). The development of a task-oriented passivity approach requires energy evaluation in the task space. Furthermore, this approach for energy evaluation allows the designer to work in a well known spatial domain, thus better understand the behaviour of the system, where to act and how to tune the parameters of the system to achieve a more stable behaviour.

Theorem 4.2.1. Energy equivalence between joint and task space *The amount of energy required in task space to perform a task is equal to the one in joint space.*

The proof of this statement requires the notion of Jacobian and Static Wrench Transmission.
The Jacobian matrix $J(q)$ allows the solution to the forward instantaneous kinematics problem for a serial chain manipulator, where the total velocity of the end effector has to be found given the the position and velocitiy of all members of the chain joints

$$^{k}v_N = J(q)\dot{q} \tag{4.20}$$

where $^{k}v_N$ is the spatial velocity of the end effector expressed in the frame k, $\dot{q}(t)$ the n-dimensional vector composed of the joints rates and $J(q)$ is a $6 \times n$ matrix whose elements are, in general, non-linear function of q and is expressed relative to the same coordinate frame as the spatial velocity $^{k}v_N$ [28].
The static wrench transmission establishes the relationship between wrenches applied to the end effector and forces/torques applied to the joints. Exploiting the principle of virtual work, the relationship between wrenches applied to the end effector and force/torques applied to the joints can be shown to be

$$\tau = J^T f \tag{4.21}$$

where τ is the n-dimensional vector of applied joint forces/torques for an n-Dof manipulator and f is the spatial wrench vector

$$f = \begin{bmatrix} \mathrm{f} \\ o \end{bmatrix} \tag{4.22}$$

in which o and f are the vectors of torques and forces, respectively, applied to the end-effector, both expressed in the coordinate frame where the Jacobian is expressed. The Jacobian maps the joint rates into the spatial velocity of the end-effector, its transpose maps the wrenches applied into the end-effector into joint forces/torques [28]. The formalization for Theorem 4.2.1 is:

$$E = \int \tau^T(t)\dot{q}(t)dt = \int f^T(t)\dot{x}(t)dt \tag{4.23}$$

and can be proved as follows

Proof. Consider the formulation of energy in joint space

$$E_\tau = \int \tau^T(t)\dot{q}(t)dt \tag{4.24}$$

thus we can write the elementary work in joint space as

$$dW_\tau = \tau^T(t)dq \quad . \tag{4.25}$$

The energy in task space is

$$E_\mathrm{f} = \int f^T(t)\dot{x}(t)dt \tag{4.26}$$

and the elementary work in task space is

$$dW_\mathrm{f} = f^T(t)dx \quad . \tag{4.27}$$

Using (4.20) we end up will

$$dW_f = f^T(t)J(q)dq \tag{4.28}$$

Manipulators are time independent system with holonomic constraints: their pose depends only on joints positions, meaning that virtual displacements match physicals ones.
By this consideration we can write

$$\delta W_\tau = \tau^T(t)\delta q \tag{4.29}$$

$$\delta W_f = f^T(t)J(q)\delta q \tag{4.30}$$

and for the principle of virtual works

$$\delta W_\mathrm{f} = \delta W_\tau \quad \forall \delta q \tag{4.31}$$

Now using (4.21) in (4.28)

$$\begin{aligned} dW_\mathrm{f} &= \mathrm{f}^T(t)J(q)dq \\ &= \tau^T(t)J^T(q)J(q)dq \\ &= \tau^T(t)dq \\ &= dW_\tau \end{aligned} \tag{4.32}$$

□

For a redundant manipulator in (4.21) the Jacobian pseudoinverse has to be addressed instead of J^T.

4.2.2 Energy scaling

When dealing with a pair of the similes manipulator, jointly coupled, performing the commanded task at the slave side will require the same amount of energy supplied at the master side. This is not true when dealing with manipulators that are mechanically different. Performing the same task require different amount of energy, even if the Cartesian motion is exactly the same. This is due to the different dynamical model of the two manipulators. Furthermore manipulator friction is another energy dissipative unknown variable to deal with. When adopting an energy based approach for enforcing the passivity of the interconnected teleoperation system, all these effects produce an energy leakage.

With the purpose to balance the different energy requirement between the two manipulators, we introduce a couple of constants α and γ such that

$$\alpha\gamma = 1 \qquad \alpha, \gamma > 0 \tag{4.33}$$

When performing the energy transmission between the two tanks, the master, which have a low inertia, send to the slave, which have a greater inertia, a properly scaled up quantum of energy to compensate the energy requirement between the two manipulator. The slave do the same but scaling down the value of the quantum of energy. In other words, if we set in e.g. $\alpha = 10$, then $\gamma = \frac{1}{10}$, it means that an action performed on the master requires ten times more energy to be replicated on the slave, at the same time a feedback from the slave require one tenth of the original energy to be executed on the master.
This idea clearly affects the energy transmission strategy that, with the introduction of energy scaling, becomes

$$\begin{cases} \alpha {}^{m}P_{out} = \alpha \left((1 - \sigma_m) D_m + {}^{s}E_{req} \beta_m \bar{P}_m \right) = {}^{s}P_{in} \\ \gamma {}^{s}P_{out} = \gamma \left((1 - \sigma_s) D_s + {}^{m}E_{req} \beta_s \bar{P}_s \right) = {}^{m}P_{in} \end{cases} \tag{4.34}$$

This enhancement preserves the passivity of the whole teleoperation system. The proof can be shown looking at the whole architecture either from the master, or from the slave side. Interdependently of the choice, each side sees the energy at the other side proportionally scaled by its scale factor.

Proof. **Slave side**
Let focus on the slave side. The storage function that represent the energy

in the system is

$$\hat{H}_s(t) = \alpha H_m(t) + H_s(t) + \alpha T_m(t) + T_s(t) \tag{4.35}$$

Using (4.10) we obtain that

$$\begin{aligned}\dot{\hat{H}}_s(t) = &- \alpha D_m(t) + \alpha F_m^T v_m + \sigma_m \alpha D_m(t) + \alpha\left(\sigma_m {}^m P_{in} - {}^m P_{out}\right) \\ &- D_s(t) + F_s^T v_s + \sigma_s D_s(t) + \left(\sigma_s {}^s P_{in} - {}^s P_{out}\right) \\ &+ \alpha u_{t_m} y_{t_m} + \alpha F_h^T v_m + u_{t_s} y_{t_s} + F_e^T v_s\end{aligned} \tag{4.36}$$

Since the tanks and the robots are interconnected with the power preserving interconnection (4.11), we have that $F_i^T v_i = -u_{t_i} y_{t_i}$, where $i = m, s$. Substituting (4.34) we have

$$\begin{aligned}\dot{\hat{H}}_s(t) = &- (1-\sigma_m)\alpha D_m(t) - (1-\sigma_s) D_s(t) \\ &\sigma_m \alpha \gamma^s P_{out} - {}^s P_{out} + \sigma_s \alpha^m P_{out} - \alpha^m P_{out} \\ &+ \alpha F_h^T v_m + F_e^T v_s\end{aligned} \tag{4.37}$$

Rearranging terms, we end up with

$$\begin{aligned}\dot{\hat{H}}^s(t) = &- (1-\sigma_m) D_m(t) - (1-\sigma_s) D_s(t) \\ &- (1-\sigma_m)^s P_{out} - (1-\sigma_s)\alpha^m P_{out} \\ &+ \alpha F_h^T v_m + F_e^T v_s\end{aligned} \tag{4.38}$$

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

$$\dot{\hat{H}}_s(t) \leq \alpha F_h^T v_m + F_e^T v_s \quad . \tag{4.39}$$

Thus (3.32) holds, which proves our statement. □

The same could be done from the master point of view by considering the storage function $\hat{H}_m(t)$

$$\hat{H}_m(t) = H_m(t) + \gamma H_s(t) + T_m(t) + \gamma T_s(t) \tag{4.40}$$

instead of (4.35).

4.3 Implemented teleoperation schemes

The effectiveness of this approach has been evaluated with two different control schemes implemented within the transparency level: a position-position (PP), for bilateral motion synchronism and a position-force (PF) for better force reflection.

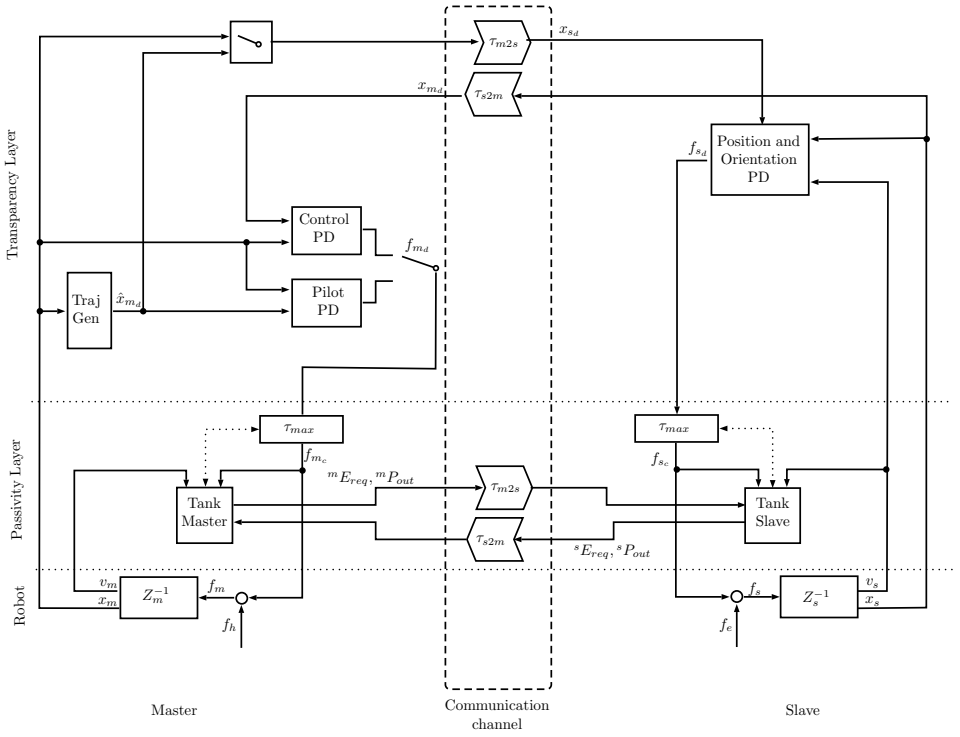


Figure 4.1: The implemented position-position architecture

4.3.1 Position - Position (P-P)

As shown in Figure 4.1, the first teleoperation schema implemented in the transparency layer is a simple Position-Position. The main goal of such schema is to achieve a synchronism between the two manipulator poses.
Master pose is sent to the slave on which a PD controller is in charge of tracking the commanded Cartesian position and orientation. Slave pose is sent back to the master and used as a feedback signal that tells the operator if the slave robot is moving in free space or if it is in contact with the environment. This is possible thanks to the virtual spring-damper coupling operated by the PD controller. In fact the virtual displacement between the two manipulator produces a force acting at both side.
When the slave is in free motion, at master side the error between the desired pose, imposed by the operator, and the feedback one is small. The slave can freely follow the pose commanded by the master, reducing the tracking error. The operator feels this little tracking displacement as a sort of inertia.

When the slave is in contact, the tracking displacement increases. At master side the error between the desired pose and the feedback one also increases, providing a stronger reaction force that acts as a kinaesthetic feedback.
Because of the P-P architecture the error displacement is the same at both sides, thus the kinaesthetic feedback induced at master side is mainly proportional to the force being applied at master side through the relationship between $K_{p_{master}}$ and $K_{p_{slave}}$. The remaining force is due to the presence of a damping factor K_d acting on the error derivative.

The slave PD pose controller is made by two separate PD controllers that works in parallel. The former acts on the Cartesian position and performs a linear control on the position error in task space. Its outputs are the control Cartesian forces, expressed in the slave's base frame, commanded to the robot.
The position error in task space is

$$e_{pos} = (x_{pos_d} - x_{pos}) \tag{4.41}$$

where x_{pos_d} is the vector of Cartesian position commanded by the master and x_{pos} the vector of current slave position
The position PD controller is formulated as

$$\mathrm{f}(t) = K_{p_{pos}} e_{pos}(t) + K_{d_{pos}} \frac{de_{pos}(t)}{dt} \tag{4.42}$$

where $K_{p_{pos}}$ and $K_{d_{pos}}$ are 3×3 diagonal matrices of the position control gains. The latter acts on Cartesian orientation and performs a linear control action on the orientation error in task space. Its output are the control Cartesian torques, expressed in the slave's base frame, commanded to the robot. The orientation error in task space is

$$e_{ang} = (x_{ang_d} - x_{ang}) \tag{4.43}$$

where x_{ang_d} and x_{ang} are the Cartesian orientation commanded by the master and the current slave orientation respectively.
The orientation PD controller is formulated as

$$o = K_{p_{ang}} e_{ang}(t) + K_{d_{ang}} v_{ang}(t) \tag{4.44}$$

where $K_{p_{ang}}$ and $K_{d_{ang}}$ are 3×3 diagonal matrices of the position control gains, and $v_{ang}(t)$ is the slave's Cartesian angular velocity vector. The

stronger damping action on the orientation controller helps the system stability.
The two components of the Cartesian wrench command, forces f and torques o are then passivated by the passivity layer (τ_{max}) and then mapped into joints torque through the Jacobian Pseudo-inverse. This mapping is performed by the robot itself.

At master side, when performing the standard bilateral P-P teleoperation, acts the same PD controller presented for the slave, tuned with proper values for $K_{p_{pos}}$, $K_{d_{pos}}$, $K_{p_{ang}}$ and $K_{d_{ang}}$.

Before performing the puncturing, the clinician could switch from this controller to a specialized one, previously mentioned as *pilot mode*, designed to help the operator to perform a more precise puncture along the needle's z axis direction.
A trajectory generator holds the values of the master orientation and $x - y$ position relative to the end-effector reference frame at switching time. The z position, instead, is always the current one.
This new conditioned command, $\hat{x}$, is provided both as a reference for the slave and as a reference value for the local PD controller.
In order to provide a different kind of feedback when in this mode, the PD controller is replicated thus we could use different gains.
The operator feels a constrain force that guides him moving only along z, in this direction the coupling between the master and the slave is preserved together with the feedback.
This controller switching ability is clearly a non passive solution. However we can safely apply such strategy because of the Two-Layer architecture guarantees the passivity of the system.

4.3.2 Position - Force (P-F)

The latter teleoperation scheme implemented is a simple Position-Force, an illustration is available in Figure 4.2, The main goal of such scheme is providing a more effective force feedback to the operator.

Master pose is sent to the slave on which, the same PD controller of the previous scheme is in charge to track the commanded Cartesian position and orientation. The feedback signal f_{slave} from the slave is the interaction wrench provided by a force sensor. At the master side this signal could be scaled down by the K_f gain which is, again, a 6×6 diagonal matrix of

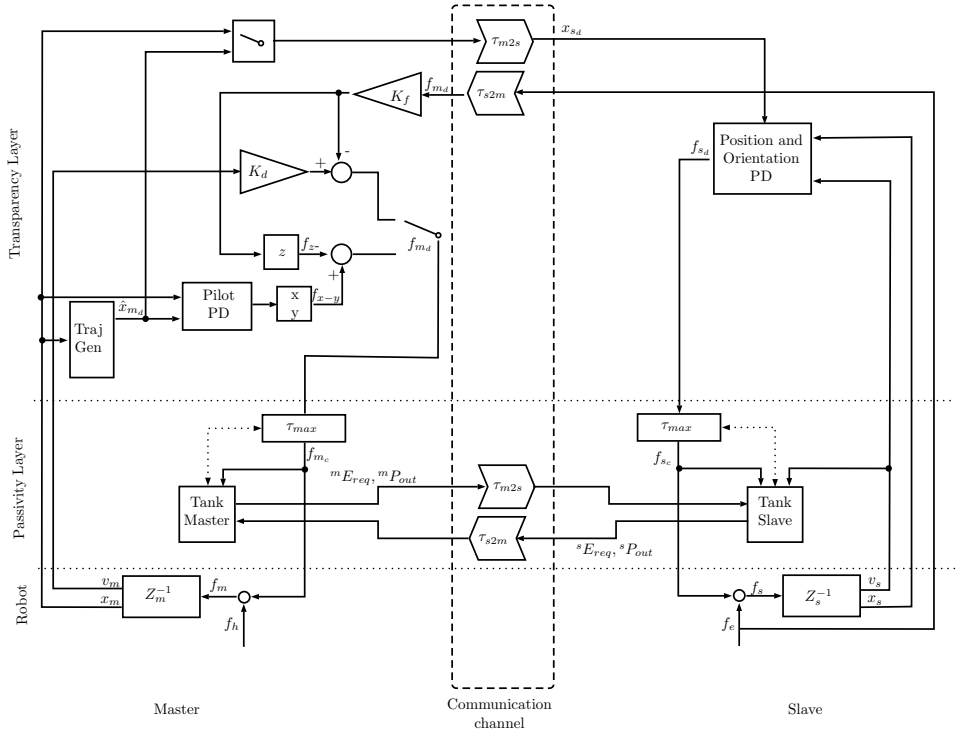


Figure 4.2: The implemented position-force architecture

gains.
Because of the two layer algorithm, we have to produce some amount of energy to perform actions at slave side while in free motion (if there are no interaction forces, the commanded force on the master is null and consequently the instantaneous power is zero). This is achieved introducing a small and constant damping. The wrench applied to the master and thus perceived by the operator is

$$\mathrm{f}(t) = K_f f_{slave}(t) - K_d v_m(t) \tag{4.45}$$

where K_d is a 6×6 diagonal matrix of damping gains and v_m is the master velocity.

Even in this architecture, the clinician before performing the puncturing could switch to the specialized controller. The purpose is the same from the P-P architecture: provide a tool to help the operator to perform a more precise puncture along the needle's z axis direction.
To achieve this goal the *pilot mode* for this schema is a hybrid position-force controller. It acts as PD controller for orientation and for the position along $x - y$ axes while keeping the original slave force feedback along z.
The trajectory generator holds the values of the master orientation and $x - y$ position relative to the end-effector frame of reference at switching time. This conditioned signal, $\hat{x}$, is provided both as a reference for the slave and as reference value for the local PD controller. The operator feels a constrain force that forces him moving only along the master tool z axis, in this direction the operator feels the corresponding force feedback from f_{slave} modulated trough K_f gain. This controller switching ability is clearly a non passive solution. However we can safely apply such strategy because of the Two-Layer architecture guarantees the passivity of the system.

4.4 Discussion

In this chapter with the Two-Layer teleoperation architecture modelled as a port-Hamiltonian system, we showed that energy evaluation is equivalent in task and in joint space. A proper energy scaling strategy has been proposed to allow the coupling, in a passive way, of very different manipulators. Finally we described the two teleoperation schemas implemented upon this formulation of Two-Layer algorithm.

Chapter 5

Experimental Results

In this chapter we present the experimental setup on which the proposed architecture has been developed, implemented and tested. First we will present the master and slave robots, then the general architecture is explained and finally we discuss on the impact that the limits of such setup have on the implementation.

5.1 Manipulators

This bilateral teleoperation setup is based upon two of the manipulators available in the Altair laboratory of the University of Verona. The master console is a Sensable Phantom Omni (recently rebranded as 3D-System Touch). The slave device is a Barret WAM in the 7-DoF configuration.

Setup	Odom VOVD	Track Pipe	Decompression	Frame Elaboration	Feature Extraction	Publish Pose	FPS Track	FPS Feature Extraction	FPS Supported
orb(j1)	-	43,41	13,08	49,21	43,62	0,02	23,04	20,32	20,32
vovd(j1)	1,70	-	-	-	-	-	-	-	-
(orb;vovd)(j1)	2,58	73,49	18,69	62,70	54,76	0,17	13,61	15,95	13,61
(orb(j1);vovd(j2))	1,60	49,81	13,53	49,25	43,01	0,08	20,08	20,30	20,08

Table 5.1: Performance Measurement: camera 10 Hz

Setup	Odom VOVD	Track Pipe	Decompression	Frame Elaboration	Feature Extraction	Publish Pose	FPS Track	FPS Feature Extraction	FPS Supported
orb(j1)	-	54,82	14,47	49,40	44,35	0,02	18,24	20,24	18,24
vovd(j1)	1,70	-	-	-	-	-	-	-	-
(orb;vovd)(j1)	2,85	93,15	23,96	67,99	60,41	0,21	10,74	14,71	10,74
(orb(j1);vovd(j2))	1,60	56,20	14,00	49,32	43,95	0,09	17,79	20,27	17,79

Table 5.2: Performance Measurement: camera 20 Hz

Setup	Odom VOVD	Track Pipe	Decompression	Frame Elaboration	Feature Extraction	Publish Pose	FPS Track	FPS Feature Extraction	FPS Supported
orb(j1)	-	42,98	35,43	82,05	55,70	0,01	23,27	12,19	12,19
vovd(j1)	1,23	-	-	-	-	-	-	-	-
(orb;vovd)(j1)	1,40	66,13	30,09	69,10	63,75	0,16	15,12	14,47	14,47
(orb(j1);vovd(j2))	1,24	46,62	30,10	74,16	67,80	0,52	21,45	13,48	13,48

Table 5.3: kubeedge - Performance Measurement: camera 10 Hz

Setup	Odom VOVD	Track Pipe	Decompression	Frame Elaboration	Feature Extraction	Publish Pose	FPS Track	FPS Feature Extraction	FPS Supported
orb(j1)	-	38,56	32,65	80,31	55,48	0,01	25,93	12,45	12,45
vovd(j1)	1,21	-	-	-	-	-	-	-	-
(orb;vovd)(j1)	1,43	74,00	28,48	68,90	63,11	0,15	13,51	14,51	13,51
(orb(j1);vovd(j2))	1,26	51,40	27,33	66,86	62,50	0,23	19,46	14,96	14,96

Table 5.4: Kubeedge - Performance Measurement: camera 20 Hz

5.1.1 Phantom Omni

The Phantom Omni is a commercial, portable haptic device with six Degrees of Freedom (DoF) with the three translational degrees of freedom actuated, developed by Sensable Technologies. It is based on a serial architecture, which means that the handle is connected to the housing by a single serial chain. The device evolved from research done by Thomas Massie and Dr. Kenneth Salisbury at MIT.

The workspace of the Phantom Omni is 16cm $\times$ 12cm $\times$ 7 cm (W $\times$ H $\times$ D) and can provide a peak force feedback up to 3.3 N. Thanks to its six DoFs and a nominal position resolution of around 0.055 mm the device is used in various professional environments. The Phantom Omni is currently sold by 3D-System under the product name Touch. The models available in the lab are the firewire and USB ones, both lacks of driver support for modern Linux distributions. Thus a Windows machine is a mandatory requirement to adopt the Phantom Omni in this setup.

The device ships with its SDK, OpenHaptics, designed for a simple control of the device. It provides two abstraction levels: namely HU and HD. The former it's oriented to provides haptic feedbacks from a virtual graphics scenario, the latter is close tight to the hardware and provides the Phantom's status information like stylus pose, stylus velocity and pressed buttons.
The available data vary upon the availability of actuated DoF on device in use: for the Omni model the API does not provide either angular velocity or the evaluated geometric Jacobian. The pose frame of reference, which is not clearly addressed in the documentation, has been empirically found in the middle of the Cartesian workspace. Finally, this is another unclear point from the available documentation, the pose provided by the HD API refers not to the stylus tip, but to the middle point of the gimbal. These

(a) The Phantom Omni

(b) The WAM available in Altair Laboratory while performin a puncturing test

Figure 5.1: The two manipulators

are useful information when dealing with the virtual model of the device. A custom version of the urdf model [2] enables the correct visualization of the pose from HD API into the RVIZ visualizer.

5.1.2 WAM

The robotic arm used in this setup as slave robot is called WAM, from Barrett Techonology. The WAM is a manipulator available in two main configurations, 4 and 7-DoF, both with human-like kinematics. The manipulator available at the Altair laboratory has the seven degrees of freedom configuration, which offers better adaptability and dexterity and is shown in Figure 5.1b.
The uniqueness of the WAM arm lies in its backdriven cable drives, similar to human tendons. This kind of design concentrates the weight at its base and makes the whole arm light enough to have little brake time together with high acceleration and flexibility. Moreover, its lightness translates into power saving. Table 5.5 its the *Denavit-Hartenberg* parameters of the Barrett Wam's arm.

For the purpose of this work, a needle with its holder has been mounted at the end effector. The assembly includes a force sensor, the ATI Nano 17, that provides the external interaction forces in the Position - Force teleoperation schema. The whole assembly, from its base to the needle's tip, could be seen as a unique static transformation that could be added at the end of

	θ	d	a	α
	θ_1	0	0	$-\pi/2$
	θ_2	0	0	$\pi/2$
	θ_3	0,55	0,045	$-\pi/2$
Robot	θ_4	0	-0,045	$\pi/2$
	θ_5	0,3	0	$-\pi/2$
	θ_6	0	0	$\pi/2$
	θ_7	0,06	0	0
End-Effector	$-0,16\pi$	0,085	0,08	0

Table 5.5: D-H parameters of the WAM manipulator with needle holder as end-effector

the WAM's kinematic chain (the last line in Table 5.5). Figure 5.2 shows the WAM end-effector.

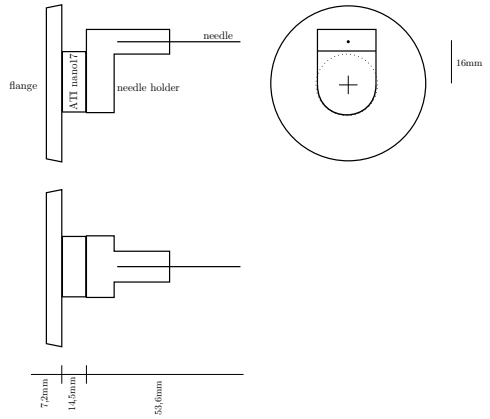
In this thesis we are focusing on the feasibility of the teleoperation architecture thus we are not interest on dealing with needle bending. Because of that, we applied to the needle holder a nail that represent an acceptable trade-off for our intent. However, with a different needle holder shown in Figure 5.3 that has been designed in the meanwhile, in future we will be able to evaluate the system with a real crioablation needle.

5.1.3 ATI Nano 17

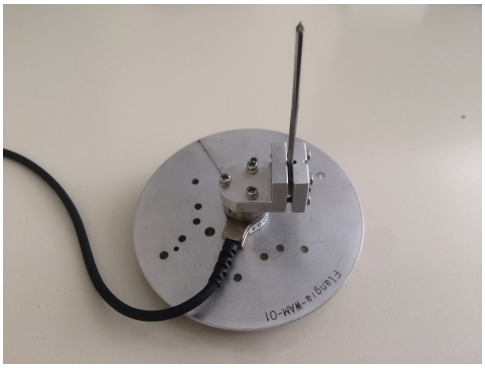
The ATI Nano 17 is the force sensor attached between the WAM tool flange and the needle holder. It's a small six axis force/torque sensor with silicon strain gages. The one available in Altair Laboratory has the SI-50-0.5 calibration whose sensing range and resolution has been reported in Table 5.6. By supplying the transformation between the sensor and the needle tip, the software in its data acquisition *NetBox* provides the measured wrenches correctly applied at the needle tip frame.

5.2 Software Architecture

The teleoperation schema has been implemented on three machines: two Linux machine and a Windows machine. The communication relays on the



(a) Schematics of the whole needle holder assembly



(b) The end-effector

Figure 5.2: The needle holder assembly

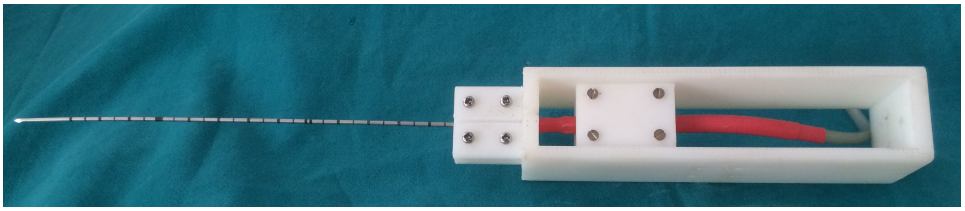


Figure 5.3: The crioablation needle holder

	Sensing ranges	Resolution
F_x	50N	1/80N
F_y	50N	1/80N
F_z	70N	1/80N
T_x	500Nmm	1/16Nmm
T_y	500Nmm	1/16Nmm
T_z	70Nmm	1/16Nmm

Table 5.6: ATI Nano 17 SI-50-0.5 calibration data

Message type	**Name**
std_msgs/Header	header
geometry_msgs/Pose	pose
geometry_msgs/Twist	twist
geometry_msgs/Wrench	wrench
std_msgs/Float64	energy_request
std_msgs/Float64	power_send

Table 5.7: The custom ROS message structure

ROS framework and on a custom made socket. The socket acts as a bridge between the ROS environment and the windows machine. Two nodes that control the two manipulators, implementing respectively the master's and slave's Two-Layer controller, are connected through a couple of network simulator node. In the middle a *frame mapper* node maps the frame reference between the two manipulators.
The communication between the nodes relays on a custom ROS message which embeds all the information required both by the Two-Layer algorithm and the teleoperation strategy implemented at the transparency level. The message structure is reported in Table 5.7 while the software architecture is sketched in Figure 5.4. Finally in Figure 5.5 we show an overview of the ROS nodes and topics involved. They will be explained within this section.

5.2.1 ROS

The Robot Operating System (ROS) [3] is a open-source framework based on the component-based software engineering paradigm that provides the middleware for inter-process communication. Initially developed by the Stanford Artificial Intelligence Laboratory its development continued at Willow Garage, a robotics research institute, and now it's maintained and improved under the action of the ORSF foundation [3].

As a meta-operating system, ROS offers features such as hardware abstraction, low-level device control, implementation of commonly used functionalities, message communication between processes and package management. It uses an asyncronouse publish/subscribing mechanism made

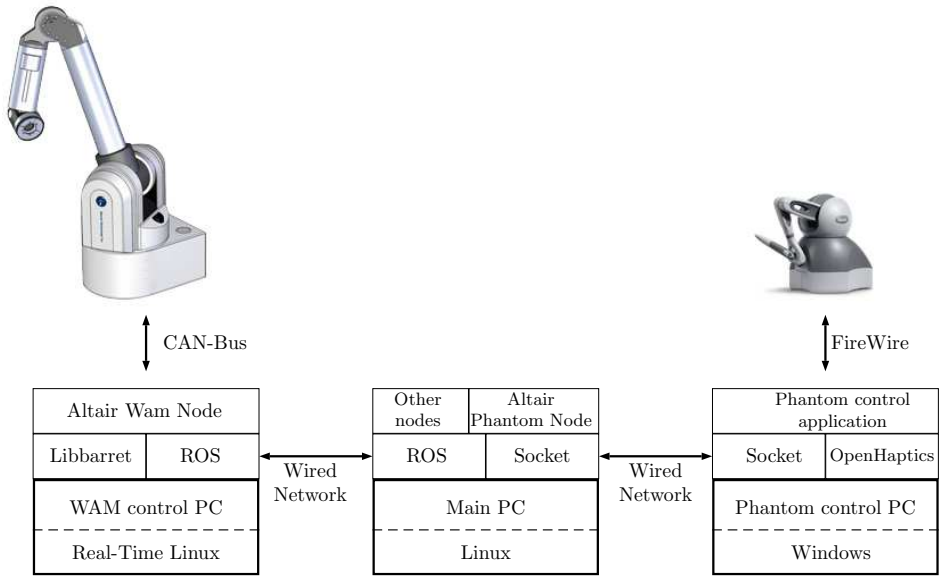


Figure 5.4: Hardware/Software overview of the teleoperation architecture

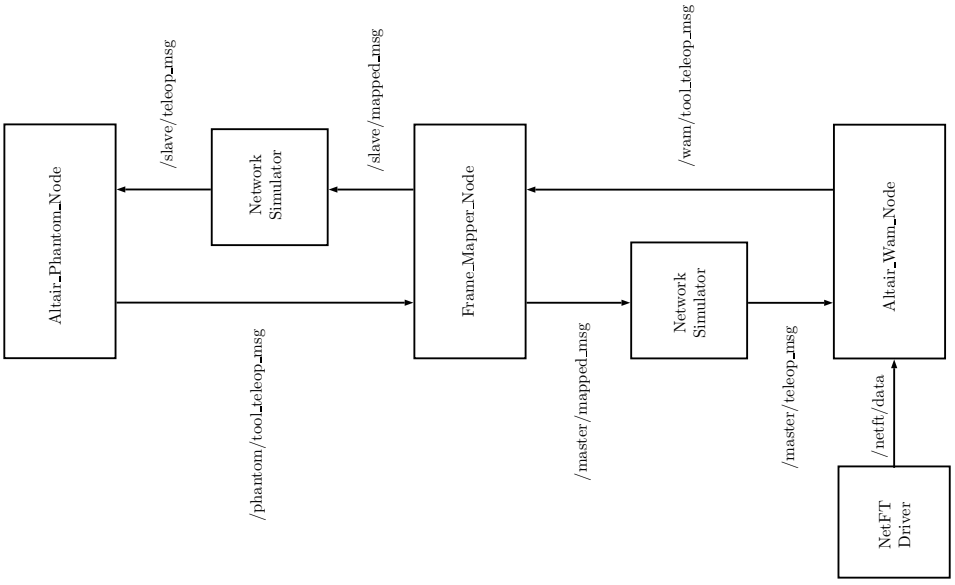


Figure 5.5: Nodes and topics in the implemented architecture

possible by message standardisation and incapsulation that make the external interface of every node as general as possible, allowing quick nodes exchange and, thus, great architectural flexibility. Each indipendent block, called *node*, executes a particular task of a process and can communicates with other nodes through *topics*. This allows to create complex architectures by aggregating many simpler entities and simplifies the use of different tasks or different methods for the same task.

Additionally to the message-passing system, the core ROS component, called *roscore*, maintains a global execution time for the nodes to achieve synchronisation. Each node executes separately with its own internal clock driven by the set execution rate. At every message sent/received, containing the internal time information, the core component updates the global time by following the execution status provided by these messages.

5.2.2 Frame Mapper

The frame mapper node is fully transparent to the application and acts by matching both pose and force reference between the two manipulators. To keep the developed architecture and each node inside it as general and modular as possible with the purpose of reusability of the developed master and slave nodes, we develop this node in a such a way it could match any pair of manipulators by providing, in its configuration, the transformation between the master and the slave base frame.
When launched, the node computes the difference between the two initial poses, then applies the proper transformation in both directions.
It could operate with any custom message type, the only requirements is the definition of a pose message inside the custom generated message.

5.2.3 WAM control

The WAM control node (`Altair_Wam_Node`) runs on the first machine, which execute the Ubuntu Linux 14.04.1 distribution patched with real time Xenomai co-Kernel. This node is build on top of the Libbarret SDK. It continuously streams the status of the robot to the ROS infrastructure by publishing a set of topics with ROS standard messages. It provides the following informations:

- **End-effector position** referred both to the base and the tool frame

of reference.

- **End-effector twist**.
- **End-effector wrench**.
- **Joints position**.

It offers some convenient ROS services to exploit simple actions like autonomous reaching of a desired configuration or the ability of idleing the robot (means the robot is gravity compensated and free to move by hand). On the basis of the ROS message received, the node engages the proper controller among the implemented ones.
This is possible by sending the control message to the proper control topic from the set of the topics the node subscribe itself. There are eight controller available right now: Cartesian position, Cartesian orientation, Cartesian pose (which is the combination of the two previous ones), Cartesian velocity, Cartesian wrench, joint position, joint velocity and finally the Two-Layer controller developed for this work.
The Two-Layer controller provides the implementation of the passivity layer, as described in Section 4.2, with the only difference that each threshold is designed as a range (a bottom and top thresholds) without violating the constraints between these variables, as explained in Section 4.10.
The transparency level implements the control strategy. For this work it's a position PD controller, which is the same for the two teleoperation schemas tested and implemented. It also provides the ability to filter the incoming control position, in order to smooth the control action when the input signal is noisy, as it happen with the measurements received from the Phantom Omni. The output of the node, the passivated control torque in task space, is passed to the underlying Libbarret real-time subsystem in charge of the low level control of the robot.

5.2.4 Phantom Omni control

Because of the lack of support for the modern Linux kernel, either for the old firewire model, and for the latest USB model, the Phantom Omni is physically attached and controlled on a Windows machine. The interface with the ROS environment is ensured through a custom designed network socket connection.

The Phantom control node (`Altair_Phantom_Node`) runs on another Linux machine, and it could establish a bidirectional communication with the Windows machine up to 1kHz. The design of the control node provides an abstraction of the Phantom manipulator in such a way that the socket interconnection could be easily replaced by a native device communication, through the OpenHaptics API, when the proper device driver will be available. In fact on the Windows machine, the software simply receives commands from the remote controller and send back the Phantom status.

On top of this device abstraction, a general, abstract *Phantom controller* has been designed. It's a partially implemented class which defines the standard *Phantom controller* interface and implements a common thread that streams to the ROS network the device status in the same way the WAM control node does. With this design, each specialization of the Phantom controller could be simply loaded into the Phantom ROS node and managed inside it in a standard way. For this work, two specializations of the Phantom Controller have been written: one for the position-position architecture and one for the position-force architecture. These specializations share the passivity layer implementation and most of the controller design with the WAM control node.

5.2.5 Network Simulator

This node has the purpose to simulate a real network environment.
It could provide constant and time-varying delay with packet losses. A couple of this node provides the communication delay in both direction.

5.2.6 Neft node

This node simply reads the wrench value from the ATI Nano 17 NetBox and streams it to the ROS network.

5.2.7 Virtual visualizer

The setup includes a virtual visualizer, based upon RVIZ, where a model of both the manipulators could be seen in a 3D environment. This tool is useful for a pre-operative check on the correctness of the transformation applied to the commanded pose for each manipulator.

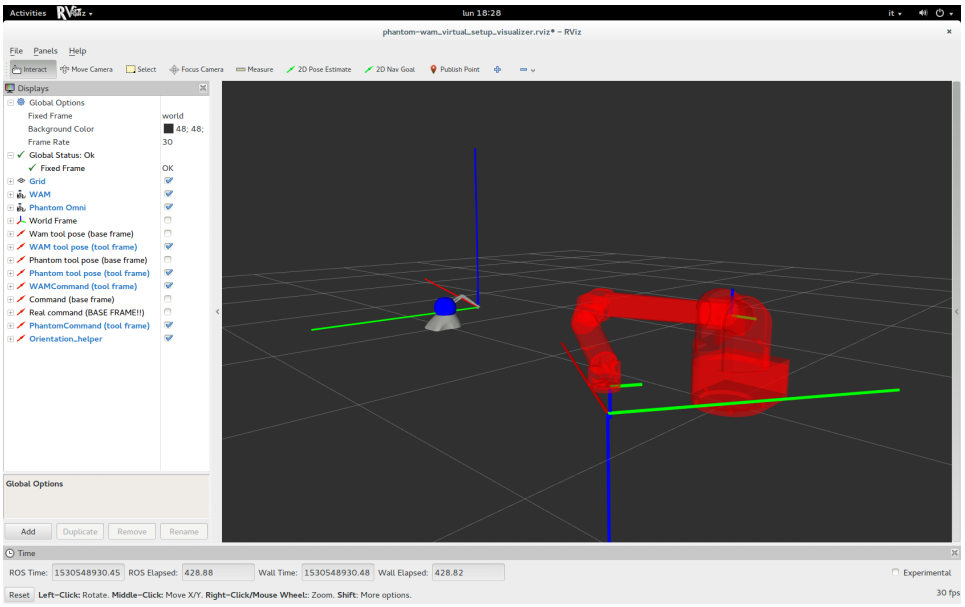


Figure 5.6: Screenshot of the visualizer in action

5.2.8 Setup limitations

Because of the only tree translational degrees of freedom actuated on the Phantom Omni, this setup implements only a translational bilateral coupling. This means that also the energy computation, both at master and slave sides, is performed only on translation.
The energy evaluated is the one flowing from the controller to the robot. The lack of actuation means that no torque is exerted on the master and consequentially the instantaneous power is always equal to zero.
Because we cannot control the orientations, for them we have to relay on a unilateral teleoperation schema.
Unfortunately we cannot measure the real interaction force with the operator, but only the commanded one. However when the force on one axes exceeds the maximum values that the Phantom Omni could render, its APIs crops the value to 3.3N. This limits the transparency of the system so we set K_f looking for a trade off to achieve a good transparency, meaning that the operator is still able to distinguish the insertion stage from the free motion.

Conclusions And Future Works

In this thesis a teleoperation architecture has been presented to address the issue of providing kinaesthetic feedback in remote percutaneous procedures such biopsy.
The inclusion of a virtual guiding mode for a more precise insertion is only one of the possible solution for augmenting the performance and exploiting such a system.
For such kind of system the problem of delay is usually not addressed due to the presence of a dedicated connection between the master console and the slave robot. The proposed teleoperated architecture has been tested with positive results even in a constant delayed network.
A future, extension of this work could be the test on a time-vary delayed network with packet loss and/or packet retransmission.
For an evaluation of the setup architecture more closer to th clinical case, the crioablation needle could be mounted and tested: this will require a re-tuning and possibly a re-shaping of the slave controller.
The substitution of the master console with a 6-DoF model, as the Phantom Premium 1.5, will enable the full potentiality of the implemented architecture with no extra effort.
With this physical change, a future work could be the evaluation of the system performance in a full bilateral configuration.
Another future improvement could be to take into account the dynamical model of the two manipulators, expressed in task space. This should reduce the need for scaling the energy between them and could enable a more suitable control strategy at the slave side for the puncturing task such as the impedance control.

Bibliography

[1] Tensorflow lite. `https://www.tensorflow.org/lite`.

[2] www.github.com/fsuarez6/phantom_omni.

[3] www.ros.org.

[4] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[5] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2016.

[6] Robert J. Anderson and Mark W. Spong. Bilateral Control of Teleoperators with Time Delay. *IEEE Transactions on Automatic Control*, 34(5):494–501, 1989.

[7] NVIDIA Corporation. Cuda zone. `https://developer.nvidia.com/cuda-zone`.

[8] NVIDIA Corporation. cuDNN deep neural network library. `https://developer.nvidia.com/cudnn`.

[9] Federica Ferraguti, Nicola Preda, Auralius Manurung, Marcello Bonfe, Olivier Lambercy, Roger Gassert, Riccardo Muradore, Paolo Fiorini, and Cristian Secchi. An Energy Tank-Based Interactive Control Architecture for Autonomous and Teleoperated Robotic Surgery. *IEEE Transactions on Robotics*, 31(5):1073–1088, 2015.

[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] Andreas Geiger. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, page 3354–3361, USA, 2012. IEEE Computer Society.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[13] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[14] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. Speed/accuracy trade-offs for modern convolutional object detectors, 2016.

[15] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Dar-

rell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia*, MM '14, page 675–678, New York, NY, USA, 2014. Association for Computing Machinery.

[16] Dale A. Lawrence. Stability and Transparency in Bilateral Teleoperation. *IEEE Transactions on Robotics and Automation*, 9(5):624–637, 1993.

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

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

[19] Rogelio Lozano, Bernard Brogliato, Olav Egeland, and Bernhard Maschke. *Dissipative Systems Analysis and Control*. Springer-Verlag London, 2000.

[20] Zongqing Lu, Swati Rallapalli, Kevin Chan, and Thomas La Porta. Modeling the resource requirements of convolutional neural networks on mobile devices. In *Proceedings of the 25th ACM International Conference on Multimedia*, MM '17, page 1663–1671, New York, NY, USA, 2017. Association for Computing Machinery.

[21] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA, 2002. Association for Computational Linguistics.

[22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach et al., editor, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[23] Xukan Ran, Haolianz Chen, Xiaodan Zhu, Zhenming Liu, and Jiasi Chen. Deepdecision: A mobile deep learning framework for edge video analytics. pages 1421–1429, 04 2018.

[24] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

[25] Sebastian Ruder. An overview of gradient descent optimization algorithms., 2016. cite arxiv:1609.04747Comment: Added derivations of AdaMax and Nadam.

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[27] D. D. Bloisi S. Aldegheri, N. Bombieri and A. Farinelli. Data flow orb-slam for real-time performance on embedded gpu boards. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5370–5375, 2019.

[28] Bruno Siciliano and Oussama Khatib. *Springer Handbook of Robotics*. Springer-Verlag, Berlin, Heidelberg, 2007.

[29] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile, 2018.

[30] Ben Taylor, Vicent Sanz Marco, Willy Wolff, Yehia Elkhatib, and Zheng Wang. Adaptive selection of deep learning models on embedded systems, 2018.