University of Cassino and Southern Lazio

Doctoral Course in
Methods, models and technologies for engineering
Curriculum in Computer Engineering
Cycle XXXII

Planning and Control of Underwater Vehicle-Manipulator Systems

---

SSD: ING-INF/04

Supervisor:
Prof. Gianluca Antonelli

Author:
Daniele Di Vito

Coordinator:
Prof.ssa Wilma Polini

"*My talent is getting things to work that people think are many decades in the future. I say we can make them happen now. "*

Rodney Brooks

Abstract

Robotics is increasingly attracting interest for more and more applications. In particular, underwater robotics is having a large development since the need of mineral resources is growing. The latter pushes to find out new deposits on the seabed. Therefore, the construction and the maintenance of underwater structures are necessary, e.g., submarine pipelines for carrying oil and gas. However, performing any task on the seabed is very dangerous for the man, for obvious reasons. Thus, several Autonomous Underwater Vehicles (AUVs), equipped with manipulators as well, have been implemented in recent years, aimed at physically substituting the man.

The control of Underwater Vehicles-Manipulator Systems (UVMSs) require full control of the vehicle. Indeed, cruise vehicles with rudder and stern are not suitable for carrying a manipulator since they are not able to counteract the interaction forces with the arm itself. Furthermore, for a rigid body moving in a fluid there exist several hydrodynamic effects acting on it. In particular, among the latter, the restoring generalized forces, which are gravity and buoyancy, and the ocean current are of major concern in designing the control law since they influence the steady-state position and orientation errors. Beyond Proportional-Integral-Derivative actions (PID), several adaptive control laws have been proposed in literature for compensating these effects. However, they all are designed starting from the dynamic models written either in the earth-fixed or in the vehicle-fixed frame, respectively. Nevertheless, some hydrodynamic terms are constant in earth-fixed frame, e.g., the restoring linear force, and some others are constant in the vehicle-fixed frame, e.g., the restoring moment. Thus, in this thesis work, a mixed earth/vehicle-fixed frame-based adaptive control able to build each dynamic compensation action in the proper reference frame is proposed. In particular, a reduced version has been derived

within the aim to achieve null steady state error under modelling uncertainty and presence of ocean current with respect to a minimal number of parameters. Furthermore, the effects of including the thruster dynamics within the full-dimensional adaptive control are investigated. Simulations and comparisons with other control laws, such as PID, show the better performance of the proposed technique.

The proposed adaptive control law has been also tested within the EC-funded ROBUST Project with the interuniversity Center of Integrated Systems for the Marine Environment (ISME). The ROBUST system has been designed and implemented for performing sea bed material identification merging the capabilities of an AUV and a robotic manipulator with a LIBS (Laser Induced Breakdown Spectroscopy) sensor mounted on its end-effector. Thus, the adaptive technique has been used for the vehicle dynamic control.

Redundant systems can be exploited to perform multiple tasks simultaneously. For this purpose, the Multi-Task Priority (MTP) inverse kinematics algorithm can be used. As well known in literature, the latter is based on the Closed Loop Inverse Kinematics (CLIK) and allows to manage a prioritized hierarchy of *equality-based* tasks, which are control functions characterized by a specific desired value, e.g, position and orientation. In addition to the latter there exists another category of control functions which are named *set-based* tasks since their value can range between an upper and lower bound, respectively. One of the most common set-based tasks is for instance the obstacle avoidance. Indeed, the obstacle distance has to respect a lower bound (a minimum distance value). However, it can assume values greater than the latter.

Within the aim to control systems taking into account safety-related tasks, it is necessary to manage both equality and set-based tasks. Thus, the Set-Based Task-Priority Inverse Kinematics (SBTPIK) is proposed in this thesis. In particular, simulations and experiments with fixed and mobile base manipulators show the effectiveness of the algorithm as well as its integration into an assistive control framework for Remotely Operated Vehicles (ROVs).

The SBTPIK validity is also demonstrated through its application within the EC-funded DexROV Project with ISME. More in detail, a via-satellite remotely

controlled UVMS has been developed to perform several kind of tasks such as oil and gas pipelines maintenance. Therefore, it has been necessary to manage multiple tasks ensuring the safety system. Thus, the SBTPIK has been applied to fulfill this objective.

The SBTPIK is a local motion control algorithm which efficiently performs on redundant systems since it handles real-time changes in the environment. However, it is prone to local minimum as any local motion controllers. Motion planners, on the other hand, are global methods and they are able to take into consideration the same system constraints. Nevertheless, their implementation often requires sacrificing some of the constraints or the redundancy exploitation. For this reason, in this thesis an approach based on merging the global and local planners in an effort to preserve the features of both ones is proposed. In particular, the global planner is implemented as a sampling-based algorithm which works in the reduced-dimensionality of the robot work space applying the Cartesian constraints only. The output trajectory is then checked against the inverse kinematics algorithm verifying the fulfillment of the other task constraints. The SBTPIK is then used also in real-time to ensure a reactive behaviour. During the movement, the motion planner runs in background to adapt to changes in the environment, human presence or, in general, to continuously optimize the path. The proposed method has been simulated within the DexROV and ROBUST frameworks and experimentally validated in laboratory with a mockup represented by the Kinova Jaco^2 7 DOFs (Degrees of Freedom) manipulator.

The SBTPIK framework has been successfully used in assistive applications as well, aimed at allowing users with severe motion disabilities to perform manipulation tasks that may help in daily-life operations. Tests have been performed using the Kinova Jaco^2 7 DOFs manipulator operated via a P300-based Brain Computer Interface (BCI). More in detail, the P300 paradigm is based on the P300 potential which is a component of the Event Related Potentials (ERPs), i.e., a fluctuation in the EEG generated by the electrophysiological response to a significant sensorial stimulus or event. In particular, the P300 consists of a positive shift in the EEG signal approximately 300-400ms after a task relevant stimulus. Thus, the user with motion disabilities can generate command

through a proper P300-based Graphical Interface (GUI). It is worth noticing that the present thesis focuses on underwater robotics therefore the BCI topic is not discussed in this work.

Contents

List of Figures

List of Tables

List of Abbreviations

AUV	Autonomous Underwater Vehicle
UVMS	Underwater Vehicle Manipulator System
ROV	Remotel Operated Vehicle
ECI	Earth-Centered-Inertial
ECEF	Earth-Centered-Earth-Fixed
NED	North-East-Down
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
UTM	Universal Transverse Mercator
CG	Center of Gravity
TCM	Thruster Configuration Matrix (TCM)
DOF	Degree Of Freedom
CLIK	Closed-Loop Inverse Kinematics
MTP	Multi-Task Priority
STPIK	Set-based Task-Priority Inverse Kinematics
DLS	Damped Least-Squares
DH	Denavit-Hartenberg
IK	Inverse Kinematics
ROS	Robotic Operating System
BCI	Brain-Computer Interface
EEG	Electroencephalographic
GUI	Graphical User Interface
CNN	Convolutional Neural Network
MBES	Multi Beam Echo Sounder
LIBS	Laser Induced Breakdown Spectroscopy
SVD	Singular Value Decomposition

Chapter 1

Introduction

1.1 Motivation

Underwater robotics is an engineering field that attracts a lot of interest for several applicative scenarios. This interest is not recent but there has always been. Famous scientists of the past devoted their efforts in designing an underwater vehicle. Leonardo Da Vinci is one of them. Indeed, a draw has been found in the *Codex Atlanticus* about a mechanical submarine (see Fig. 1.1). According to legends, Leonardo worked on the project of an underwater military machine which then he destroyed since judged too dangerous.

Figure 1.1: Render of the mechanical submarine designed by Leonardo Da Vinci.

In the last years, underwater robotics has known an increasing development due to the growing need of mineral resources, that pushes to find out new deposits on the seabed, and to the need to construct and to do maintenance of underwater structures, e.g., submarine pipelines used primarily to carry oil and gas. Indeed, the aim of the research is to perform these tasks in a completely

autonomous way. Therefore, through the years, increasingly advanced systems have been developed to physically substitute the human in performing these tasks because of the dangerous conditions. However, performing autonomous underwater tasks is still challenging both from the technological and theoretical point of view since it involves a wide range of technical and research topics. The underwater environment is actually unknown and unstructured with poor visibility in most of cases. Then, the vision based systems are generally not reliable. Furthermore, the online communication is limited since technologies as GNSS (Global Navigation Satellite System) are not applicable due to the impossibility of underwater electromagnetic transmission at GNSS specific frequencies. Therefore, all these factors require some on board *intelligence* and the ability to react in a proper way according to the specific scenario.

The actuation system of an underwater vehicle is strictly related to the type of mission to perform. Cruise vehicles are usually equipped with one thruster and several control surfaces. Furthermore, they are usually under-actuated and controlled in surge, sway and heave directions. However, for vehicles aimed at manipulation tasks, that are the focus of this work, the actuation in all DOFs is necessary, therefore requiring configurations with 6 or more thrusters.

Thrusters are propellers generally driven by electrical motors. They can be divided into two groups: thrusters and tunnel thrusters. The latter consist in propellers mounted in ducts or shrouds which increase their static and dynamic efficiency. Furthermore thrusters are usually asymmetric and optimized for producing thrust force in one direction; on the other hand, tunnel thrusters are usually symmetric [70, 63]. Thus most small-to-medium-sized underwater vehicles are powered by tunnel-thrusters.

Underwater robots are mainly divided into two types: Remotely Operated Vehicles (ROVs) and Autonomous Underwater Vehicles (AUVs). ROVs are underwater systems physically linked via tether to a human operator whose working station can be either on a submarine or on a surface vessel, as shown in Fig. 1.2. The tether is a crucial element for this kind of vehicle since it is in charge of giving power to the ROV as well as closing the manned control loop, i.e., by providing sensor information to the operator and by transmitting motion directives to the ROV. However, its presence makes more difficult the navigation

and manoeuvring tasks; indeed, the operator has to reduce the twisting and to avoid entanglements while taking into account the tether inertia effects on the vehicle. Thus man continues to have an essential role.

Figure 1.2: ROV and pilots from Comex, France, within the European Community funded DexROV Project.

AUVs, on the other hand, are systems supposed to be completely autonomous with onboard power and unmanned control loop. Some tests have shown that the use of an AUV in place of a ROV for the same task can have a cost saving up to 85% [4]. Furthermore, the tether absence, for the reasons mentioned above, represents a consistent advantage especially for survey missions. Therefore, AUVs are knowing an increasing interest, in particular from the research perspective. However, the presence of technological issues, first among all the limited autonomy of the power system, makes them not widely used inside industry that prefers to rely on ROVs or on a combined use of both of them [45, 32]. In case of missions which require manipulation tasks, e.g., turning a valve, the vehicle can be equipped with one or more manipulators. Then, it is an Underwater Vehicle-Manipulator System (UVMS), as the system shown in Fig. 1.3.

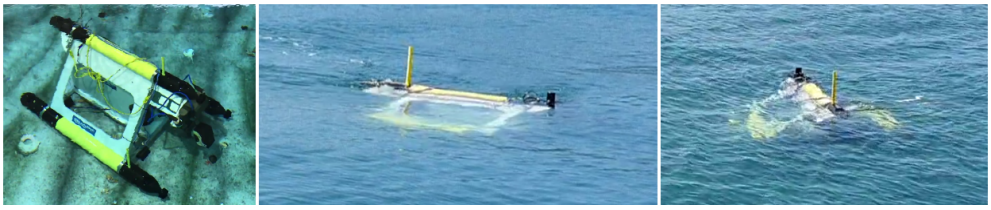

Figure 1.3: UVMS developed within the European Community funded ROBUST Project.

1.2 UVMS's Control

Control of UVMSs is challenging since the underwater environment makes the dynamics of a six Degrees of Freedom (DOFs) rigid body quite different from industrial and service robotics. Indeed, when a rigid body is moving in a fluid the latter is accelerated by its movement. Therefore, the additional inertia of the fluid surrounding the body has to be considered. Nevertheless, this effect can be neglected in the industrial and service robotics since the air density is much lighter than the one of the moving mechanical system. However, in underwater scenarios, the water density is comparable to the vehicle one.

The overall Control framework for a UVMS carrying a n-DOFs arm is represented in Fig. 1.4. In particular, the motion planner is the highest Control level and it is in charge of planning the motion given a desired end-effector pose $\boldsymbol{\eta}_{ee,d} \in \mathbb{R}^6$. The resulting trajectory $\boldsymbol{r}(t)$ is sent to the Kinematic layer which tracks the desired reference generating the proper system velocities $\boldsymbol{\zeta}_d \in \mathbb{R}^{6+n}$. The latter are sent to the Dynamic layer that is the lowest Control level which computes the required torques $\boldsymbol{\tau}_c$ for the velocity tracking. Then, the resulting torques are sent to the vehicle and applied to the motors. Every control layer takes the vehicle pose $\boldsymbol{\eta} \in \mathbb{R}^6$ and the arm joint positions $\boldsymbol{q} \in \mathbb{R}^n$ as feedback.

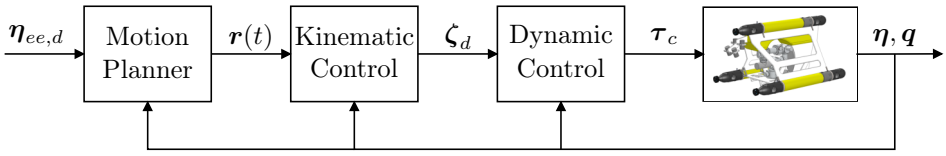

Figure 1.4: Overall control architecture for an UVMS.

1.3 Thesis outline and contribution

This thesis work focuses on the architecture control of an underwater vehicle-manipulator system. Indeed, the three main control layers are analysed: the dynamic control, the kinematic control and the motion planner. For each control layer the background is presented and new methods are proposed showing

the numerical/experimental validation. In particular, this thesis is divided in the following chapters:

- **Chapter 2 - Background**: this chapter focuses on the background and, therefore, on the state of the art regarding the control of UVMSs. After a brief introduction on the reference systems used in underwater robotics, the dynamic model with the thruster allocation are presented. Then, the major drawbacks of the state of the art dynamic control techniques are described. Regarding the kinematic layer, the Multi-Task Priority (MTP) inverse kinematics is presented putting in evidence its limit, namely the possibility to control only equality-based tasks. Then, the motion planner is introduced describing the drawbacks regarding both the global and local planners.
- **Chapter 3 - Dynamic Control**: an adaptive control technique able to properly compensate for hydrodynamic effects is proposed in this chapter. Furthermore, a recursive adaptive approach is proposed to counteract the arm interactions. An analysis of thruster dynamics and its effects on the control loop is conducted as well. Numerical simulations are reported to validate the adaptive control taking into consideration both a AUV and a UVMS. Furthermore, the EU funded ROBUST project is presented reporting the comparison results between an adaptive and PI (Proporional-Integral) control performed within this framework.
- **Chapter 4 - Kinematic Control**: this chapter focuses on the kinematic control. More in detail, the Set-based Task-Priority Inverse Kinematics (STPIK), that is a framework able to manage both equality-based and set-based tasks, is presented. Furthermore, aimed at properly manage kinematic singularities, a deep analysis on Damped Least Square (DLS) Algorithms is conducted as well. The effectiveness of the proposed inverse kinematics controller is then shown in the simulation of an assistive framework ROV piloting, which has been implemented taking into account needs of real ROV pilots, and proved by its application within the EU funded DexROV project. Indeed, the DexROV system has required several safety-related tasks which have been performed through the STPIK algorithm.

- **Chapter 5 - Motion Planner**: the focus of this chapter is the motion planning. Indeed, an approach based on merging a global and local planner, in an effort to preserve the features of both ones, is presented. In particular, the STPIK is used as local planner, since it is a reactive method able to fulfill the system and environment constraints. The effectiveness of this approach is experimentally validated using as mockup a 7 DOFs fixed-based manipulator.

The activities described in this thesis work led to publication of the following papers (ordered from the most recent to the oldest one):

- P. Di Lillo, D. Di Vito, G. Antonelli, *"Set-Based Inverse Kinematics control of an UVMS within the DexROV project"*, OCEANS 2018 - Charleston, IEEE, 2018;
- D. Di Vito, E. Cataldi, P. Di Lillo, G. Antonelli, *"Vehicle Adaptive Control for Underwater Intervention Including Thrusters Dynamics"*, IEEE Control Technology and Applications (CCTA), 2018.
- D. Di Vito, E. Cataldi, P. Di Lillo, G. Antonelli, *"Assistive Control Framework for Remotely Operated Vehicles"*, Mediterranean Conference on Control and Automation (MED), IEEE, 2018;
- P. Di Lillo, D. Di Vito, E. Simetti, G. Casalino, G. Antonelli, *"Satellite-based tele-operation of an underwater vehicle-manipulator system. Preliminary experimental results"*, IEEE International Conference on Robotics and Automation (ICRA), 2018;
- D. Di Vito, G. Antonelli, *"The effect of the ocean current in the thrusters closed-loop performance for underwater intervention"*, OCEANS 2017 - Anchorage, IEEE, 2017.
- D. Di Vito, C. Natale, G. Antonelli, *"A Comparison of Damped Least Squares Algorithms for Inverse Kinematics of Robot Manipulators"*, IFAC-PapersOnLine, 2017;

and to the submission of the following one:

- D. Di Vito, M. Bergeron, D. Meger, G. Dudek, G. Antonelli, S. Chiaverini, *"Dynamic planning of redundant robots within a framework of set-basedtask-priority inverse kinematics"*, submitted to IEEE International Conference on Robotics and Automation (ICRA), 2020.

Furthermore, the kinematic control algorithm proposed in the present thesis work has been applied in BCI-based assistive applications leading to the following publication:

- F. Arrichiello, P. Di Lillo, D. Di Vito, G. Antonelli, S. Chiaverini, *"Assistive Robot operated via a Brain-Computer Interface"*, IEEE International Conference on Robotics and Automation (ICRA), 2017;

and to the following submissions:

- P. Di Lillo, F. Arrichiello, D. Di Vito, and G. Antonelli, *"BCI-controlled assistive manipulator: developed architecture and experimental results"*, revise and re-submit, IEEE Transactions on Cognitive and Developmental Systems, 2019.
- G. Gillini, P. Di Lillo, F. Arrichiello, D. Di Vito, A. Marino, G. Antonelli and S. Chiaverini, *"A dual arm mobile robot system performing assistive tasks operated via P300-based Brain Computer Interface"*, revise and re-submit, IEEE Robotics and Automation Magazine (RAM), 2019.

Chapter 2

Background

2.1 Reference System

In underwater robotics the pose of a vehicle can be expressed with respect to Earth-centered coordinate frames, such as the Earth-Centered-Inertial (ECI) and the Earth-Centered-Earth-Fixed (ECEF) ones, or other geographic reference frames such as the North-East-Down (NED) one.

The ECI frame has its origin in the earth center with x, y axis on the equator plane and z pointing to the north pole, respectively. It is assumed inertial since it is not subject to accelerations.

The ECEF frame has its origin in the center of earth with x, y axis on the equator and z towards north pole as well. However, differently from the ECI one, the latter rotates together with the earth (see Fig. 2.1). Thus, the x axis always points to the intersection between the equator and the prime meridian (latitude 0°, longitude 0°).

The NED frame is obtained by defining a local plane tangent to the area of interest. This plane has the x and the y-axis pointing to the north and east, respectively. The z-axis completes the frame (see Fig. 2.2). It is worth noticing that in this coordinate system the earth is approximated with an ellipsoid according to the World Geodetic System standard. Then, the vector z does not point to the earth center. Furthermore, the NED frame origin is fixed but initialized by the user according to their needs.

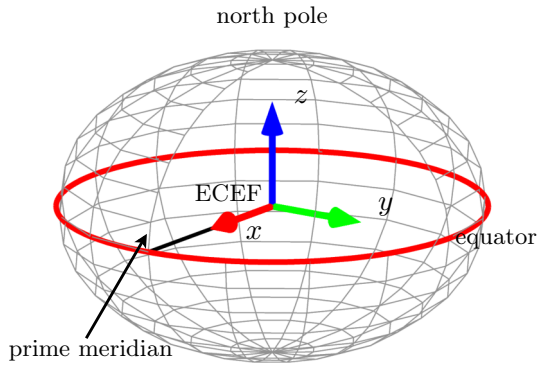

Figure 2.1: Earth-Centered-Earth-Fixed (ECEF) frame rotating together the earth.

The GNSSs, e.g., the American Global Positioning System (GPS), can provide coordinates both in ECEF frame and in the Universal Transverse Mercator (UTM) system. The latter is a kind of NED frame system. In particular, it provides the point localization on the earth surface projecting the latter on a plane and therefore ignoring the altitude.

In underwater control, the most used coordinate system to express the vehicle pose is the NED frame convention. Thus, henceforth the NED frame is referred to as Inertial frame Σ_I.

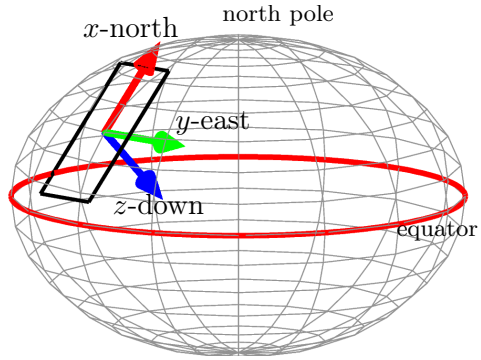

Figure 2.2: North-East-Down (NED) frame with the origin defined by user.

The Body frame Σ_B is fixed on the vehicle under study, as shown in Fig. 2.3. The x axis is usually aligned with the vehicle forward direction (from aft to

fore). The z axis points down and y axis completes the frame. Its origin can be placed in any point of the vehicle depending on the specific needs. In most of cases, it coincides with the Center of Gravity (CG).

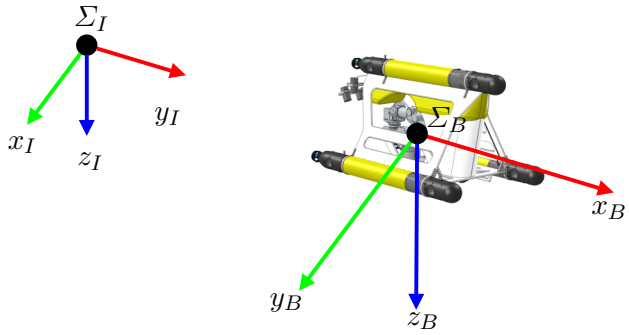

Figure 2.3: Inertial Σ_I and Body Σ_B frames representation.

2.1.1 SNAME Notation

The common notation used in underwater robotics according to the SNAME notation [1] is reported in table 2.1. Figure 2.4 shows the reference frames and

		Forces and Moments	velocity	pose
Motion in the x-direction	Surge	X	u	x
Motion in the y-direction	Sway	Y	v	y
Motion in the z-direction	Heave	Z	w	y
Motion in the x-direction	Roll	K	p	ϕ
Motion in the y-direction	Pitch	M	q	θ
Motion in the z-direction	Yaw	N	r	ψ

Table 2.1: SNAME notation for Marine Robotics.

elementary motions according to this convention. In detail,

- $[X\ Y\ Z]^{\mathrm{T}}$ are the *forces* in Newton ([N]);
- $[K\ M\ N]^{\mathrm{T}}$ are the *moments/torques* in Newton·meter ([Nm]);
- $[u\ v\ w]^{\mathrm{T}}$ are the *linear velocities* in meter/second ([m/s]);
- $[p\ q\ r]^{\mathrm{T}}$ are the *angular velocities* in radiant/second ([rad/s]) or degree/second ([deg/s]);

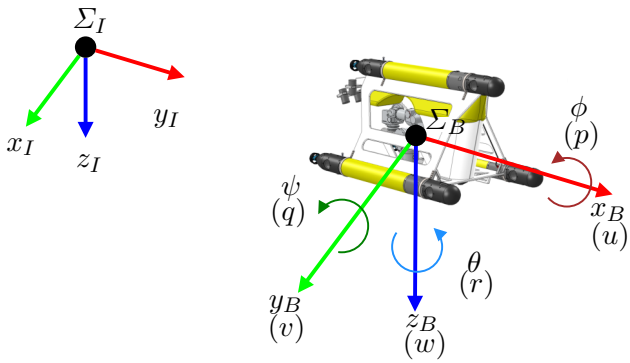

Figure 2.4: Inertial Σ_I and Body Σ_B frames representation with elementary vehicle's motion.

- $[x\ y\ z]^{\mathrm{T}}$ are the *position components* in meter ([m]);
- $[\phi\ \theta\ \psi]^{\mathrm{T}}$ are the *orientation components* (angles) in radiant ([rad]) or degree ([deg]).

2.2 Vehicle Pose

An underwater vehicle can be considered as a rigid body. Thus, its pose with respect to an inertial frame Σ_I is defined as

$$\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{\eta}_1 & \boldsymbol{\eta}_2 \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^6 \tag{2.1}$$

where $\boldsymbol{\eta}_1 = \begin{bmatrix} x & y & z \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^3$ and $\boldsymbol{\eta}_2 = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^3$ are the position and orientation vectors, respectively. In particular, the vehicle orientation can be represented in several ways.

2.2.1 Roll-Pitch-Yaw attitude

One way to represent the vehicle attitude is the Euler Angle convention:

$$\boldsymbol{\eta}_2 = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^3\,, \tag{2.2}$$

where ϕ, θ, ψ are the roll, pith and yaw angles, respectively. The corresponding rotation matrix, following the zyx-convention in current frame, is

$$\boldsymbol{R}_B^I = \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\psi s_\theta s_\phi & s_\psi s_\phi + c_\psi c_\phi s_\theta \\ s_\psi c_\theta & c_\psi c_\phi + s_\phi s_\theta s_\psi & -c_\psi s_\phi + s_\theta s_\psi c_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \tag{2.3}$$

where $c_* = \cos(*)$ and $s_* = \sin(*)$. This kind of attitude representation is affected by the so called *singularity representation*. This means that the conversion from the rotation matrix to the RPY representation is not defined for $\theta = \pm\frac{\pi}{2}$ rad. The representation singularity does not represent an issue for underwater vehicles since the latters usually work with roll and pitch close to zero especially for energetic reasons. However, taking into consideration the UVMS, it is not possible to guarantee small values for pitch angles. Since the representation singularity affects any three-parameters attitude representation, it is necessary to use a non-minimal one such as the Unit Quaternion.

2.2.2 Unit Quaternion

The Unit Quaternion is a non minimal four parameters attitude representation. More in detail, let the orientation between two frames with same origin be defined as

$$\boldsymbol{R}_k(\alpha) = \cos\alpha \boldsymbol{I}_{3\times 3} + (1 - \cos\alpha)\boldsymbol{k}\boldsymbol{k}^{\mathrm{T}} - \sin\alpha \boldsymbol{S}(\boldsymbol{k}) \tag{2.4}$$

where k is the unit vector identifying the axis which has to be rotated of the α angle to align the two frames, $\boldsymbol{I}_{3\times 3}$ is the identity matrix and $\boldsymbol{S}$ is the skew matrix performing the cross product between two (3×1) vectors. Then, the unit quaternion is defined as

$$\mathcal{Q}(\eta, \boldsymbol{\varepsilon}) \tag{2.5}$$

where

$$\eta = \cos\frac{\alpha}{2}, \quad \boldsymbol{\varepsilon} = \boldsymbol{k}\sin\frac{\alpha}{2} \tag{2.6}$$

are the scalar and vector part, respectively, satisfying the following condition:

$$\eta^2 + \boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\varepsilon} = 1 \; . \tag{2.7}$$

Furthermore, since the two quaternions $\mathcal{Q}(\eta, \boldsymbol{\varepsilon})$ and $\mathcal{Q}(-\eta, -\boldsymbol{\varepsilon})$ represent the same orientation, aimed at guaranteeing the representation uniqueness, in Eq. (2.6) it is assumed $\eta \geq 0$ that corresponds to $\alpha \in [-\pi; \pi]$. Given the Unit Quaternion the corresponding rotation matrix is

$$\boldsymbol{R}(\mathcal{Q}) = \begin{bmatrix} 2\big(\eta^2 + \varepsilon_x^2\big) - 1 & 2\big(\varepsilon_x\varepsilon_y - \eta\varepsilon_z\big) & 2\big(\varepsilon_x\varepsilon_z + \eta\varepsilon_y\big) \\ 2\big(\varepsilon_x\varepsilon_y + \eta\varepsilon_z\big) & 2\big(\eta^2 + \varepsilon_y^2\big) - 1 & 2\big(\varepsilon_y\varepsilon_z - \eta\varepsilon_x\big) \\ 2\big(\varepsilon_x\varepsilon_z - \eta\varepsilon_y\big) & 2\big(\varepsilon_y\varepsilon_z + \eta\varepsilon_x\big) & 2\big(\eta^2 + \varepsilon_z^2\big) - 1 \end{bmatrix} \; . \tag{2.8}$$

2.2.3 Vehicle Velocities

The linear velocity of the body fixed frame Σ_B with respect to inertial frame Σ_I expressed in Σ_B is

$$\boldsymbol{\nu}_1 = [u \;\; v \;\; w]^{\mathrm{T}} \in \mathbb{R}^3 \tag{2.9}$$

where u, v, w are the velocities along the x, y, z-axis, respectively. In particular, the following relation exists between Σ_I and Σ_B:

$$\dot{\boldsymbol{\eta}}_1 = \boldsymbol{R}_B^I \boldsymbol{\nu}_1 \tag{2.10}$$

where $\dot{\boldsymbol{\eta}}_1$ is the time derivative of the position vector $\boldsymbol{\eta}_1$ expressed in Σ_I and $\boldsymbol{R}_B^I$ is the rotation matrix defined in Eq. (2.3).

The angular velocity of Σ_B with respect to Σ_I expressed in Σ_B is

$$\boldsymbol{\nu}_2 = [p \;\; q \;\; r]^{\mathrm{T}} \in \mathbb{R}^3 \tag{2.11}$$

where p, q, r are the velocities around the x, y, z-axis, respectively. Differently from the linear components, the time derivative of the orientation vector $\boldsymbol{\eta}_2$ as defined in Eq. (2.2) does not have a physical meaning. For this reason a proper transformation matrix $\boldsymbol{T}(\boldsymbol{\eta}_2)$ is necessary to express the relation between Σ_B

and Σ_I. More in detail, $\boldsymbol{T}(\boldsymbol{\eta}_2)$ can be derived as

$$\boldsymbol{\nu}_2 = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \boldsymbol{R}_{x,\phi}^{\mathrm{T}} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \boldsymbol{R}_{x,\phi}^{\mathrm{T}} \boldsymbol{R}_{y,\theta}^{\mathrm{T}} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} = \boldsymbol{T}^{-1}(\boldsymbol{\eta}_2)\dot{\boldsymbol{\eta}}_2 \,. \tag{2.12}$$

Therefore, the following yields:

$$\boldsymbol{T}^{-1}(\boldsymbol{\eta}_2) = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & c_\theta s_\phi \\ 0 & -s_\phi & c_\theta c_\phi \end{bmatrix}, \; \boldsymbol{T}(\boldsymbol{\eta}_2) = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi / c_\theta & c_\phi / c_\theta \end{bmatrix} \tag{2.13}$$

with $c_* = \cos(*), s_* = \sin(*), t_* = \tan(*)$ and the following relation between Σ_I and Σ_B exists:

$$\dot{\boldsymbol{\eta}}_2 = \boldsymbol{T}(\boldsymbol{\eta}_2)\boldsymbol{\nu}_2 \,. \tag{2.14}$$

2.2.4 UVMS Velocities

Aimed at studying also UVMSs, the following variable is defined:

$$\boldsymbol{\zeta} = \begin{bmatrix} \boldsymbol{\nu}_1 & \boldsymbol{\nu}_2 & \dot{\boldsymbol{q}} \end{bmatrix}^{\mathrm{T}} \,, \tag{2.15}$$

where $\dot{q} \in \mathbb{R}^n$ is the time derivative of the joint positions with n the number of joints. Then, the relation between Σ_B and Σ_I is expressed as

$$\begin{bmatrix} \dot{\boldsymbol{\eta}}_1 \\ \dot{\boldsymbol{\eta}}_2 \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_B^I & \boldsymbol{O}_{3\times 3} & \boldsymbol{O}_{3\times n} \\ \boldsymbol{O}_{3\times 3} & \boldsymbol{T}(\boldsymbol{\eta}_2) & \boldsymbol{O}_{3\times n} \\ \boldsymbol{O}_{n\times 3} & \boldsymbol{O}_{n\times 3} & \boldsymbol{I}_n \end{bmatrix} \begin{bmatrix} \boldsymbol{\nu}_1 \\ \boldsymbol{\nu}_2 \\ \dot{\boldsymbol{q}} \end{bmatrix} \,. \tag{2.16}$$

It is worth noticing that the velocity of the manipulator end-effector, defined as

$$\dot{\boldsymbol{\eta}}_{ee} = \begin{bmatrix} \dot{\boldsymbol{\eta}}_{ee,1} & \dot{\boldsymbol{\eta}}_{ee,2} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^6 \,, \tag{2.17}$$

is related to the body-fixed velocities by a proper Jacobian, as shown in Section 2.4.1.

2.3 Dynamics

The equations of motion can be derived by applying Newtonian mechanics. The dynamic model is very important since it allows the system analysis and in particular the design of model based control algorithms.

2.3.1 Dynamic Model

The dynamic model of a moving AUV in terms of body frame coordinates is [34]

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{C}_{A}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\boldsymbol{R}_I^B) = \boldsymbol{\tau}_v \tag{2.18}$$

where $\boldsymbol{M} \in \mathbb{R}^{6\times6}$ includes both the rigid body inertial matrix and the added mass effects, $\boldsymbol{C}_{RB}(\boldsymbol{\nu}) \in \mathbb{R}^{6\times6}$ is the rigid body Coriolis-centripetal matrix, $\boldsymbol{C}_{A}(\boldsymbol{\nu}) \in \mathbb{R}^{6\times6}$ is the hydrodynamic Coriolis-centripetal matrix due to the added mass effects, $\boldsymbol{D}(\boldsymbol{\nu}) = \boldsymbol{D} + \boldsymbol{D}_n(\boldsymbol{\nu}) \in \mathbb{R}^{6\times6}$ is the damping matrix containing, respectively, linear and quadratic terms, $\boldsymbol{g}(\boldsymbol{R}_I^B) \in \mathbb{R}^6$ is the vector of gravitational and buoyant forces, $\boldsymbol{\tau}_v \in \mathbb{R}^6$ is the vector of forces and moments acting on the AUV and $\boldsymbol{\nu}$ is the corresponding velocity vector in Σ_B. Within the aim to take into consideration the ocean current, the relative velocity is introduced:

$$\boldsymbol{\nu}_r = \boldsymbol{\nu} - \boldsymbol{R}_I^B\boldsymbol{\nu}_c^I \,, \tag{2.19}$$

with $\boldsymbol{\nu}_c^I$ the ocean current velocity in earth-fixed frame. Thus, the model in Eq. (2.18) can be written as

$$\boldsymbol{M}\dot{\boldsymbol{\nu}}_r + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{C}_{A}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{g}(\boldsymbol{R}_I^B) = \boldsymbol{\tau}_v \,. \tag{2.20}$$

Furthermore, the dynamic model of a rigid body can be rewritten in regressor form exploiting the linearity in the dynamic parameters:

$$\boldsymbol{Y}_v(\boldsymbol{R}_B^I, \boldsymbol{\nu}, \dot{\boldsymbol{\nu}})\boldsymbol{\theta}_v = \boldsymbol{\tau}_v \,, \tag{2.21}$$

with $\boldsymbol{Y}_v(\boldsymbol{R}_B^I, \boldsymbol{\nu}, \dot{\boldsymbol{\nu}})$, $\boldsymbol{\theta}_v$ the regressor matrix and the parameters vector of proper dimensions, respectively.

Taking into consideration a UVMS, the dynamic model defined in Eq. (2.18) is rewritten as

$$\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{\zeta}} + \boldsymbol{C}(\boldsymbol{q},\boldsymbol{\zeta})\boldsymbol{\zeta} + \boldsymbol{D}(\boldsymbol{q},\boldsymbol{\zeta})\boldsymbol{\zeta} + \boldsymbol{g}(\boldsymbol{R}_I^B) = \boldsymbol{\tau}_{vq} \tag{2.22}$$

where

$$\boldsymbol{\tau}_{vq} = \begin{bmatrix} \boldsymbol{\tau}_v \\ \boldsymbol{\tau}_q \end{bmatrix} \in \mathbb{R}^{6+n} \tag{2.23}$$

with

$$\boldsymbol{\tau}_q = \begin{bmatrix} \boldsymbol{\tau}_{q,1} \\ \vdots \\ \boldsymbol{\tau}_{q,n} \end{bmatrix} \in \mathbb{R}^n \tag{2.24}$$

the manipulator joint torques. It is straightforward to observe from Eq. (2.22) that each matrix and vector have dimensions $(6+n)\times(6+n)$ and $(6+n)$, respectively. Furthermore, the UVMS dynamic model can be rewritten in regressor form as well:

$$\boldsymbol{Y}_{vq}(\boldsymbol{q}, \boldsymbol{R}_B^I, \boldsymbol{\zeta}, \dot{\boldsymbol{\zeta}})\boldsymbol{\theta}_{vq} = \boldsymbol{\tau}_{vq} \ , \tag{2.25}$$

where $\boldsymbol{Y}_{vq} \in \mathbb{R}^{(6+n)\times n_\theta}$, with n_θ the number of parameters.

2.3.2 Actuator allocation

The relationship between forces/moments acting on the vehicle $\boldsymbol{\tau}_v$ and the control inputs to the thruster $\boldsymbol{u}_{\text{tr}} \in \mathbb{R}^p$, with p the number of propellers, is nonlinear. Indeed it depends on structural variables such as the vehicle velocity, the water density, the hydrodynamic effects of the fluid around the propeller and consequently the propeller features. However, in practice, the following relation between the vector $\boldsymbol{\tau}_v$ and the control inputs $\boldsymbol{u}_{\text{tr}}$ is used [35]:

$$\boldsymbol{\tau}_v = \boldsymbol{B}_v \boldsymbol{u}_{\text{tr}} \in \mathbb{R}^6 \tag{2.26}$$

where $\boldsymbol{B}_v \in \mathbb{R}^{6\times p}$, is the Thruster Configuration Matrix (TCM). The TCM is a known constant matrix which depends on the particular thruster allocation.

For a UVMS the relation with the control inputs is rewritten as

$$\tau_{\text{uvms}} = \begin{bmatrix} \boldsymbol{\tau}_v \\ \boldsymbol{\tau}_q \end{bmatrix} = \begin{bmatrix} \boldsymbol{B}_v & \boldsymbol{O}_{6\times n} \\ \boldsymbol{O}_{n\times 6} & \boldsymbol{I}_n \end{bmatrix} \boldsymbol{u} = \boldsymbol{B}\boldsymbol{u} \in \mathbb{R}^{6+n} \tag{2.27}$$

with $\boldsymbol{u} = \begin{bmatrix} \boldsymbol{u}_{\text{tr}} & \boldsymbol{u}_q \end{bmatrix} \in \mathbb{R}^{6+n}$ the thruster and joint control inputs, respectively. As an example, the TCM, with respect to Fig. 2.5 which represents a typical thruster configuration for a fully actuated and redundant vehicle, is:

$$\boldsymbol{B}_v = \begin{bmatrix} c_{\theta_1} & c_{\theta_2} & 0 & 0 & c_{\theta_5} & c_{\theta_6} & 0 & 0 \\ s_{\theta_1} & s_{\theta_2} & 0 & 0 & s_{\theta_5} & s_{\theta_6} & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & r_{y_3} & r_{y_4} & 0 & 0 & r_{y_7} & r_{y_8} \\ 0 & 0 & r_{x_3} & r_{x_4} & 0 & 0 & r_{x_7} & r_{x_8} \\ B_{61} & B_{62} & 0 & 0 & B_{65} & B_{66} & 0 & 0 \end{bmatrix} \tag{2.28}$$

where

$$c_{\theta_i} = \cos(\theta_i),\ s_{\theta_i} = \sin(\theta_i) \tag{2.29}$$

$$B_{6i} = r_{x_i}\sin(\theta_i) - r_{y_i}\cos(\theta_i) \tag{2.30}$$

with r_{x_i}, r_{y_i} and θ_i the i-th thruster position and orientation with respect to the vehicle-fixed frame for $i = \{1, \ldots, 8\}$, respectively.

2.3.3 Dynamic Control

UVMSs are systems aimed at performing tasks requiring some kind of interaction with the environment. Then, they have to be able to counteract the interaction force with the arm itself. Therefore, it is necessary to control the vehicle in 6 DOFs compensating the hydrodynamic effects. In particular, among the several hydrodynamic effects, the restoring generalized forces (gravity and buoyancy $\boldsymbol{g}(\boldsymbol{R}_I^B)$) and the ocean current $\boldsymbol{\nu}_c$ are of particular interest in designing the control law for underwater vehicles. Indeed, the latter, if not well compensated, can cause steady-state vehicle pose errors. These effects are more

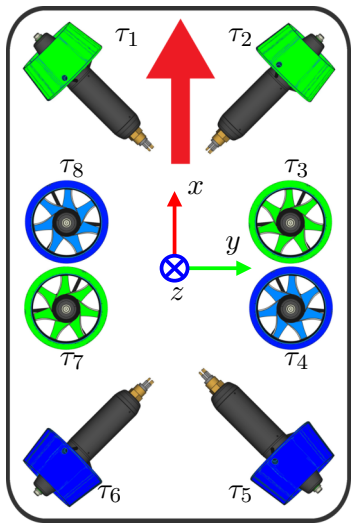

Figure 2.5: Top view of a fully actuated and redundant underwater vehicle with 8 thrusters.

evident in large dimensions vehicle since small displacements of the centers of gravity and buoyancy require major thrust efforts for compensating [50].

Several adaptive control laws are proposed in literature to compensate the hydrodynamic effects (e.g., [77, 27]). However, only the restoring forces are usually taken into consideration, whereas only few works consider the ocean current effects [33, 39, 61, 8]. Furthermore, most of the adaptive control laws are designed either in earth-fixed frame Σ_I or in body-fixed frame Σ_B. Therefore, they are not able to compensate each hydrodynamic effect with respect to the proper frame. Indeed, some effects are constant in Σ_I, e.g., the restoring linear forces, and other ones are constant in Σ_B, e.g., the restoring moment. For these reasons it is necessary a mixed earth/vehicle-fixed frame-based model-based controller able to build the compensation action in the proper reference frame as the control algorithm proposed in [6].

2.4 Kinematics

Differently from *dynamics* that considers forces causing the motion, *kinematics* takes into consideration the geometrical aspect of motion.

2.4.1 Differential Kinematics

Taking into consideration a UVMS, *kinematics* allows to establish the relation between the vehicle/joints $(\boldsymbol{\eta}, \boldsymbol{q})$ variables and the end-effector pose $\boldsymbol{\eta}_{ee}$. In particular, the direct kinematics

$$\boldsymbol{\eta}_{ee} = \boldsymbol{k}(\boldsymbol{\eta}, \boldsymbol{q}) \tag{2.31}$$

allows to determine the end-effector pose by vehicle/joints trajectories. Nevertheless, the end-effector trajectory $\boldsymbol{\eta}_{ee}(t)$ is usually given. Then, it is necessary the inverse kinematics. Since Eq. (2.31) is not linear, the differential kinematics is considered:

$$\begin{bmatrix} \dot{\boldsymbol{\eta}}_{ee1} \\ \boldsymbol{\omega}_{ee} \end{bmatrix} = \boldsymbol{J}(\boldsymbol{R}_B^I, \boldsymbol{q})\boldsymbol{\zeta} \ , \tag{2.32}$$

that maps the $(6+n)$-dimensional vehicle/joints velocities into m-dimensional end-effector velocities by the Jacobian matrix defined as [9]

$$\boldsymbol{J}(\boldsymbol{R}_B^I, \boldsymbol{q}) = \begin{bmatrix} \boldsymbol{J}_{pos}(\boldsymbol{R}_B^I, \boldsymbol{q}) & \boldsymbol{J}_{or}(\boldsymbol{R}_B^I, \boldsymbol{q}) \end{bmatrix} \in \mathbb{R}^{6\times 6+n} \tag{2.33}$$

where

$$\boldsymbol{J}_{pos}(\boldsymbol{R}_B^I, \boldsymbol{q}) = \begin{bmatrix} \boldsymbol{R}_B^I & -(\boldsymbol{S}(\boldsymbol{R}_B^I \boldsymbol{r}_{B0}^B) + \boldsymbol{S}(\boldsymbol{R}_0^I \boldsymbol{\eta}_{0,ee}^0))\boldsymbol{R}_B^I & \boldsymbol{J}_{pos,man} \end{bmatrix} \in \mathbb{R}^{3\times 6+n} \tag{2.34}$$

$$\boldsymbol{J}_{or}(\boldsymbol{R}_B^I, \boldsymbol{q}) = \begin{bmatrix} \boldsymbol{O}_{3\times 3} & \boldsymbol{R}_B^I & \boldsymbol{J}_{or,man} \end{bmatrix} \in \mathbb{R}^{3\times 6+n} \tag{2.35}$$

with $\boldsymbol{S}(\cdot)$ the skew matrix performing the cross product, $\boldsymbol{r}_{B0}^B$, $\boldsymbol{\eta}_{0,ee}^0$ the vectors connecting Σ_B (the body-fixed frame) to Σ_0 (the manipulator base frame) and Σ_0 to $\boldsymbol{\eta}_{ee}$ (the manipulator end-effector) as shown in Fig. 2.6, and $\boldsymbol{J}_{pos,man}$, $\boldsymbol{J}_{or,man}$ the manipulator position and orientation jacobian matrix, respectively.

2.4.2 Kinematic Redundancy

The UVMS is redundant if it has more DOFs than those necessary to perform the specific task, e.g., $(6+n) > m$. In this case, Eq. (2.32) admits infinite

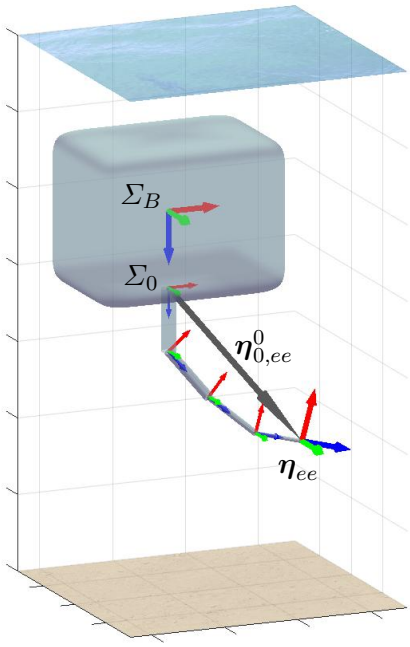

Figure 2.6: UVMS representation with in evidence the body-fixed frame Σ_B, the manipulator base frame Σ_0 and the vector $\boldsymbol{\eta}^0_{0,ee}$ connecting Σ_0 to $\boldsymbol{\eta}_{ee}$.

solutions. Therefore, the kinematic redundancy can be exploited to perform other tasks. Indeed, the end-effector configuration is not the only variable of interest, but other tasks can be accomplished simultaneously, i.e., vehicle/end-effector obstacle avoidance, vehicle attitude and joint limits.

The generic task variable is defined as

$$\boldsymbol{\sigma}_x = \boldsymbol{\sigma}_x(\boldsymbol{\eta}, \boldsymbol{q}) \in \mathbb{R}^m \tag{2.36}$$

Its differential relation is given by

$$\dot{\boldsymbol{\sigma}}_x = \boldsymbol{J}_x(\boldsymbol{\eta}, \boldsymbol{q})\boldsymbol{\zeta} \ , \tag{2.37}$$

where $\boldsymbol{J}_x(\boldsymbol{\eta}, \boldsymbol{q}) \in \mathbb{R}^{m \times 6+n}$ is the corresponding Jacobian matrix relating its time derivative $\dot{\boldsymbol{\sigma}}$ to the system velocity $\boldsymbol{\zeta} = \begin{bmatrix} \boldsymbol{\nu}_1 & \boldsymbol{\nu}_2 & \dot{\boldsymbol{q}} \end{bmatrix}^{\mathrm{T}}$.

2.4.3 Differential Inverse Kinematics

The mapping defined in Eq. (2.37) needs to be inverted for computing the system velocity corresponding to the specific task time derivative. However, the matrix $\boldsymbol{J}_x$ is not invertible since it is not square. Then, the Moore-Penrose pseudoinverse is used:

$$\boldsymbol{\zeta}_d = \boldsymbol{J}_x^{\dagger}(\boldsymbol{\eta}, \boldsymbol{q}) \dot{\boldsymbol{\sigma}}_x \tag{2.38}$$

with

$$\boldsymbol{J}_x^{\dagger}(\boldsymbol{\eta}, \boldsymbol{q}) = \boldsymbol{J}_x^{\mathrm{T}}(\boldsymbol{\eta}, \boldsymbol{q}) \big(\boldsymbol{J}_x(\boldsymbol{\eta}, \boldsymbol{q}) \boldsymbol{J}_x^{\mathrm{T}}(\boldsymbol{\eta}, \boldsymbol{q}) \big)^{-1} \tag{2.39}$$

obtaining the solution corresponding to the minimization of the vehicle/joints velocities in a least-square sense of the following quadratic cost function [66]:

$$g(\boldsymbol{\zeta}) = \frac{1}{2} \boldsymbol{\zeta}^{\mathrm{T}} \boldsymbol{\zeta} \, . \tag{2.40}$$

It is possible to use a weighted pseudoinverse as well:

$$\boldsymbol{J}_{x,\boldsymbol{W}}^{\dagger}(\boldsymbol{\eta}, \boldsymbol{q}) = \boldsymbol{W}^{-1} \boldsymbol{J}_x^{\mathrm{T}}(\boldsymbol{\eta}, \boldsymbol{q}) \big(\boldsymbol{J}_x(\boldsymbol{\eta}, \boldsymbol{q}) \boldsymbol{W}^{-1} \boldsymbol{J}_x^{\mathrm{T}}(\boldsymbol{\eta}, \boldsymbol{q}) \big)^{-1} \, . \tag{2.41}$$

The latter is useful especially for a coordinated system, such as a UVMS, since it allows to weight the vehicle and manipulator movement in a proper way. Indeed, the vehicle movement should be avoided when it is unnecessary.

The digital implementation of Eq. (2.38) leads to a numerical drift since it is based on a numerical integration procedure for obtaining vehicle/joints positions. For this reason, a closed loop version of the above equation is used, that is the Closed-Loop Inverse Kinematics (CLIK) algorithm [23]. In particular, the CLIK links the system velocity compute to the task error

$$\tilde{\boldsymbol{\sigma}}_x = \boldsymbol{\sigma}_{x,d} - \boldsymbol{\sigma}_x \, , \tag{2.42}$$

with $\boldsymbol{\sigma}_{x,d}$ the desired task value. Thus, Eq. (2.38) is rewritten as

$$\boldsymbol{\zeta}_d = \boldsymbol{J}_x^{\dagger}(\boldsymbol{\eta}, \boldsymbol{q})\big(\dot{\boldsymbol{\sigma}}_{x,d} + \boldsymbol{K}_x\tilde{\boldsymbol{\sigma}}_x\big) \tag{2.43}$$

where $\dot{\boldsymbol{\sigma}}_{x,d}$ is the desired task time derivative and $\boldsymbol{K}_x$ is a positive definite gain matrix.

Notice that the subscript d in $\boldsymbol{\zeta}_d$ means "desired value" since it is the reference velocity corresponding to the specific desired task which is sent to the system low level, as shown in Fig. 1.4.

2.4.4 Kinematic Singularities

The manipulator joint configurations at which the matrix $\boldsymbol{J}$ is rank deficient, e.g., $\text{rank}(\boldsymbol{J}) < m$, are called kinematic singularities. When the system is close to these particular configurations, a small increment in the task value can cause high joint velocities [25]. Furthermore the structure mobility is reduced.

The kinematic singularities can be classified into 2 categories: at the boundary and within the workspace. In the latter case, three singularity types can be identified: shoulder, elbow and wrist respectively.

The kinematic singularity problem can be avoided resorting to the Damped Least-Square (DLS) pseudoinverse [54] defined as

$$\boldsymbol{J}_x^{\dagger\lambda}(\boldsymbol{\eta}, \boldsymbol{q}) = \boldsymbol{J}_x^{\mathrm{T}}(\boldsymbol{\eta}, \boldsymbol{q})\big(\boldsymbol{J}_x(\boldsymbol{\eta}, \boldsymbol{q})\boldsymbol{J}_x^{\mathrm{T}}(\boldsymbol{\eta}, \boldsymbol{q}) + \lambda^2\boldsymbol{I}_m\big)^{-1} \tag{2.44}$$

where $\lambda \in \mathbb{R}$ is a damping factor and $\boldsymbol{I}_m \in \mathbb{R}^{m\times m}$ is the identity matrix.

2.4.5 Task priority redundancy resolution

The kinematic redundancy can be exploited to perform multiple tasks. In particular, given two tasks $\boldsymbol{\sigma}_a$, $\boldsymbol{\sigma}_b$ these solutions follow:

$$\boldsymbol{\zeta}_a = \boldsymbol{J}_a^{\dagger}(\boldsymbol{\eta}, \boldsymbol{q})\big(\dot{\boldsymbol{\sigma}}_{a,d} + \boldsymbol{K}_a\tilde{\boldsymbol{\sigma}}_a\big) \tag{2.45}$$

$$\boldsymbol{\zeta}_b = \boldsymbol{J}_b^{\dagger}(\boldsymbol{\eta}, \boldsymbol{q})\big(\dot{\boldsymbol{\sigma}}_{b,d} + \boldsymbol{K}_b\tilde{\boldsymbol{\sigma}}_b\big)\ . \tag{2.46}$$

Then, it is necessary to combine them in a single solution. In literature there are several approaches for managing multiple tasks. This thesis focuses on the Multi-Task Priority (MTP) inverse kinematics [47, 55] for which the velocity vector for $\boldsymbol{\sigma}_a$, $\boldsymbol{\sigma}_b$ is given by:

$$\boldsymbol{\zeta}_d = \boldsymbol{\zeta}_a + \left(\boldsymbol{J}_b \boldsymbol{N}_a\right)^{\dagger} \left(\dot{\boldsymbol{\sigma}}_{b,d} + \boldsymbol{K}_b \tilde{\boldsymbol{\sigma}}_b - \boldsymbol{J}_b \boldsymbol{\zeta}_a\right) \tag{2.47}$$

where $\boldsymbol{N}_a \in \mathbb{R}^{(6+n)\times(6+n)}$ is the null matrix projecting the system velocities corresponding to the task $\boldsymbol{\sigma}_b$ in the null space of the Jacobian $\boldsymbol{J}_a$. In this way the motion related to $\boldsymbol{\sigma}_b$ does not affect the task space $\boldsymbol{\sigma}_a$, which is then defined task with higher priority (see Fig. 2.7).

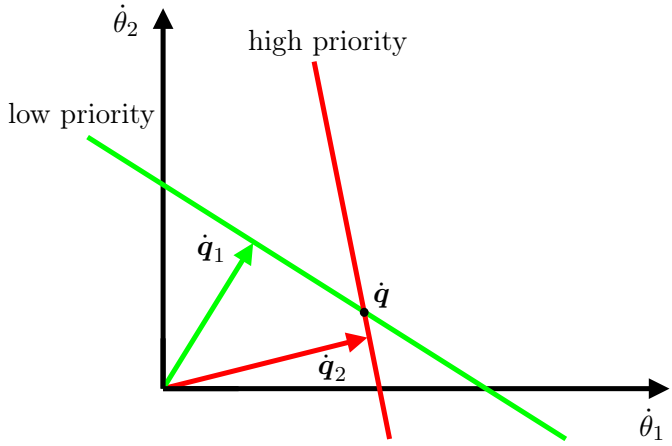

Figure 2.7: Graphical representation of the solution computed by the Multi-Task Priority (MTP) inverse kinematics for a 2 DOFs system with two compatible tasks: the red and green line are the solutions sets fulfilling the high and low-priority tasks, respectively; their intersection is the solution satisfying both ones.

However, there is a drawback in case of tasks conflict. In particular, two tasks $\boldsymbol{\sigma}_a$ and $\boldsymbol{\sigma}_b$ are in conflict if $R(\boldsymbol{J}_a^T) \cap R(\boldsymbol{J}_b^T) \neq \emptyset$ (see Fig. 2.8). When this occurs, the matrix $\boldsymbol{J}_b \boldsymbol{N}_a$ loses rank even if $\boldsymbol{J}_a$ and $\boldsymbol{J}_b$ are not rank-deficient. Then, the solution is ill-conditioned and large joint velocities can result, as in proximity of a kinematic singularity. For this reason, this event is referred to as algorithmic singularity and it can be handled through the DLS pseudoinverse as well as the kinematic singularity.

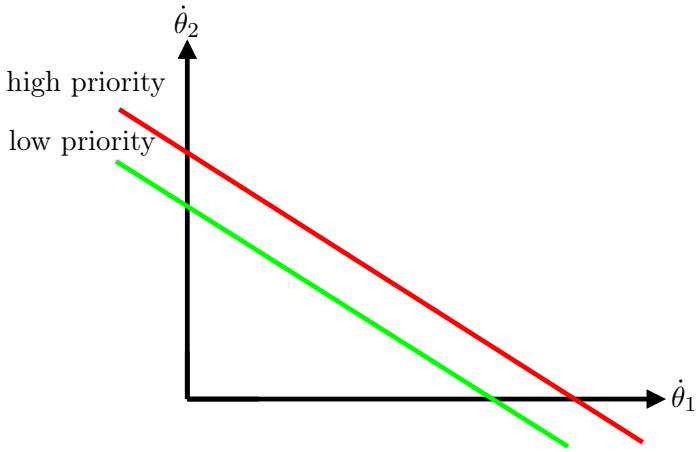

Figure 2.8: Algorithmic singularity for a 2 DOFs system: the two tasks are in conflict, therefore, the solution computed by the Multi-Task Priority (MTP) inverse kinematics is ill-conditioned and needs to be properly handled (e.g. through the DLS pseudoinverse).

The MTP algorithm can be recursively applied to high redundant systems as in [69]. More in detail, the reference system velocity can be computed as in the following way:

$$\boldsymbol{\zeta}_{d,h} = \sum_{i=1}^{h} (\boldsymbol{J}_i \boldsymbol{N}_{i-1}^A)^\dagger (\dot{\boldsymbol{\sigma}}_{i,d} + \boldsymbol{K}_i \tilde{\boldsymbol{\sigma}}_i - \boldsymbol{J}_i \boldsymbol{\zeta}_{i-1}) \,, \tag{2.48}$$

where $\boldsymbol{N}_i^A$ is the null space projector of the augmented Jacobian matrix, obtained by stacking all the task Jacobian matrices from task 1 to i

$$\boldsymbol{J}_i^A = \begin{bmatrix} \boldsymbol{J}_1^T & \boldsymbol{J}_2^T & \dots & \boldsymbol{J}_i^T \end{bmatrix}^T \,, \tag{2.49}$$

defined as:

$$\boldsymbol{N}_i^A = \boldsymbol{I} - (\boldsymbol{J}_i^A)^\dagger \boldsymbol{J}_i^A \,, \tag{2.50}$$

with $\boldsymbol{N}_0^A = \boldsymbol{I}$, $\boldsymbol{\zeta}_0 = \boldsymbol{0}$, and $\boldsymbol{\zeta}_i$ is the solution at the i-th iteration, namely the reference system velocity that fulfills all the tasks from the first one to the i-th one.

2.4.6 Set-based Control

Tasks can be classified in equality-based and set-based. Equality-based tasks have an exact desired value as control objective, e.g., the end-effector pose. On the other hand, set-based tasks, usually known as inequalities constraints, have a set of possible values, e.g., the obstacle avoidance.

Set-based tasks are commonly handled by expressing the inverse kinematics problem in a sequence of Quadratic Programming problems and solving the latter through iterative algorithms [41, 48]. Therefore, this approach can result slow since it is computationally heavy. In [67] set-based tasks are added to the task priority framework using proper activation functions during task transition to guarantee the smoothness of the output reference trajectory. Nevertheless, during transitions the task priority is lost and, therefore, the safety constraints could not be satisfied.

The MTP algorithm, introduced above, can handle only equality tasks. However, it has been extended in [51, 10, 30] to handle set-based tasks as well. Thus the Set-based Task-Priority Inverse Kinematics (STPIK) has been implemented.

2.5 Motion Planner

Unlike industrial environments which are structured and allow feed-forward behaviour, daily life scenarios present a dynamic environment. They exhibit a configuration which can change over time and can not be predicted because it often depends on the actions of nearby participants, e.g., in the human-robot cooperation and in the service robotics [65, 31]. The robotic systems obviously need to be safe, first with respect to the humans and then to preserve the machine integrity.

One possible method to accomplish the above mentioned control objectives is to properly embed them into inverse kinematics frameworks. However, being local methods, they are all prone to local minimum that can cause the specific operation to fail.

Methods based on task-constrained motion planning [44] are proposed to overcome the local minimum problem of the above mentioned reactive control techniques. Some proposed methods work in joint space [71] and they rely on particular projection approaches [13, 12] to find the system configuration which satisfies the constraints. Therefore, a proper representation of the task constraints in the joint space is necessary. However, the latter is not straightforward since the mapping between the joint space and the task space is not linear. Indeed, motion planners performing directly in the task space are proposed [64]. Furthermore, within the aim to take into consideration the closed-loop system dynamics, a control-aware motion planning algorithm based on a strict coupling between planning and control is presented in [74].

However, motion planners usually have a non-negligible computational load. In particular, the generic motion planner could not be able to re-plan in time if there is a configuration change of the environment during the system movement. Therefore, the task constraints would not be fulfilled causing very low safety conditions, especially in presence of obstacles.

For the above reasons, several methods aimed at performing fast re-plan are proposed. A real-time Model Predictive Control (MPC) with proper handling of various kind of constraints is proposed in [14]. In [52] an obstacle-avoidance circuitry for a 6-DOF Jaco robot manufactured by Kinova is built allowing its motion planning with 1 ms time. The work [57] proposes a method which plans in the task-constrained configuration space exploiting the redundancy for robotic structures in order to achieve the given task path. In [20], [56] further develops of this method allowing for planning safe cyclic motions under repetitive task constraints are shown. Moreover, the same algorithm is properly extended with the time dimension in [21], [19], in order to generate a trajectory that is compatible with moving obstacles along a known single-task trajectory, assuming to know the obstacles' trajectory.

In this thesis an approach based on the combination of a global and local planner taking advantage of both ones, aimed at performing fast re-planning, is presented.

Chapter 3

Dynamic Control

The dynamic control computes the necessary torques to perform trajectory tracking compensating for model uncertainties and persistent dynamic effects. Thus, a control law able to adapt to dynamic parameters and to consider disturbances in the proper frame has higher performances.

3.1 AUV Adaptive Control

Defining the vehicle desired pose as $\boldsymbol{\eta}_d(t) \in \mathbb{R}^6$ and the corresponding desired velocity as $\boldsymbol{\nu}_d(t) \in \mathbb{R}^6$, respectively, the following is introduced:

$$\boldsymbol{\nu}_{\text{ref}} = \boldsymbol{\nu}_d + \begin{bmatrix} \lambda_p \boldsymbol{I}_{3\times3} & \boldsymbol{O}_{3\times3} \\ \boldsymbol{O}_{3\times3} & \lambda_o \boldsymbol{I}_{3\times3} \end{bmatrix} \boldsymbol{e} \tag{3.1}$$

where

$$\boldsymbol{e} = \begin{bmatrix} \boldsymbol{R}_I^B \tilde{\boldsymbol{\eta}}_1 \\ \tilde{\boldsymbol{\varepsilon}} \end{bmatrix} \in \mathbb{R}^6 \tag{3.2}$$

is the pose error, with the orientation expressed in unit quaternion, and λ_p, λ_o are positive gains. Furthermore, the following variable is introduced:

$$\boldsymbol{s} = \boldsymbol{\nu}_{\text{ref}} - \boldsymbol{\nu} \ . \tag{3.3}$$

Thus, designing a PD-like action plus an adaptive compensation action, the control law is given by [9]

$$\boldsymbol{\tau}_c = \boldsymbol{Y}\,\hat{\boldsymbol{\theta}} + \boldsymbol{K}_s\boldsymbol{s}\ , \tag{3.4}$$

where the vector of the estimated dynamic parameters $\hat{\boldsymbol{\theta}}$ is updated according to the following relation

$$\dot{\hat{\boldsymbol{\theta}}} = \boldsymbol{K}_\theta^{-1}\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{s} \tag{3.5}$$

with $\boldsymbol{K}_s \in \mathbb{R}^{6\times 6}$ and $\boldsymbol{K}_\theta \in \mathbb{R}^{\mu\times\mu}$ positive definite design gain matrices.

3.1.1 Stability Analysis

The stability analysis proved in [8] is here reported. In detail, the vehicle dynamic model is rewritten in such a way that the inertia matrix $\boldsymbol{M}$ is put in evidence:

$$\boldsymbol{M}\boldsymbol{\nu} + \boldsymbol{n}(\boldsymbol{\nu}, \boldsymbol{R}_I^B) = \boldsymbol{\tau}_v \tag{3.6}$$

with $\boldsymbol{n}(\cdot)$ collecting all the other dynamic parameters in a 6×1 generic vector function of the system state.

Let consider the scalar function

$$V(\boldsymbol{s}, \tilde{\boldsymbol{\theta}}) = \frac{1}{2}\boldsymbol{s}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{s} + \frac{1}{2}\tilde{\boldsymbol{\theta}}^{\mathrm{T}}\boldsymbol{K}_\theta\tilde{\boldsymbol{\theta}} \tag{3.7}$$

which is positive since $\boldsymbol{M}$ and $\boldsymbol{K}_\theta$ are positive definite, with $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta} - \hat{\boldsymbol{\theta}}$ the dynamic parameter estimation error. Differentiating V with respect to time yields

$$\dot{V} = \boldsymbol{s}^{\mathrm{T}}\boldsymbol{M}(\dot{\boldsymbol{\nu}}_{\mathrm{ref}} - \dot{\boldsymbol{\nu}}) - \tilde{\boldsymbol{\theta}}^{\mathrm{T}}\boldsymbol{K}_\theta\dot{\hat{\boldsymbol{\theta}}} \tag{3.8}$$

where the parameters have been considered constant, e.g., $\dot{\tilde{\boldsymbol{\theta}}} = -\dot{\hat{\boldsymbol{\theta}}}$. Considering Eq. (3.6) yields

$$\dot{V} = \boldsymbol{s}^{\mathrm{T}}(\boldsymbol{M}\dot{\boldsymbol{\nu}}_{\mathrm{ref}} + \boldsymbol{n} - \boldsymbol{\tau}_v) - \tilde{\boldsymbol{\theta}}^{\mathrm{T}}\boldsymbol{K}_\theta\dot{\hat{\boldsymbol{\theta}}} \tag{3.9}$$

that exploiting the linearity in the parameters is

$$\dot{V} = \boldsymbol{s}^{\mathrm{T}}(\boldsymbol{Y}\,\hat{\boldsymbol{\theta}} - \boldsymbol{\tau}_v) - \tilde{\boldsymbol{\theta}}^{\mathrm{T}}\boldsymbol{K}_\theta\dot{\hat{\boldsymbol{\theta}}} \tag{3.10}$$

with $\boldsymbol{Y}$ defined in Eq. (2.25). Finally, taking into consideration Eq. (3.4, 3.5) the following is obtained:

$$\dot{V} = -\boldsymbol{s}^{\mathrm{T}} \boldsymbol{K}_s \boldsymbol{s} \tag{3.11}$$

The system stability can be now proved in a Lyapunov-like sense using the Barbălat's Lemma [42]. In particular,

since
- V is lower bounded
- $\dot{V} \leq 0$
- V is uniformly continuous

then $\dot{V} \to 0$ as $t \to 0$.

Therefore, $\boldsymbol{s} \to 0$ as $t \to 0$. Taking into consideration Eq. (3.3), it is straightforward that $\boldsymbol{s} \to 0$ implies $\boldsymbol{\nu}_d - \boldsymbol{\nu} \to 0$ and $\boldsymbol{e} \to 0$. Nevertheless, the asymptotic stability of the whole state can not be proved, as usual in adaptive control technique, since $\tilde{\boldsymbol{\theta}}$ is only guaranteed to be bounded.

3.1.2 AUV Reduced controller

The Adaptive Control law discussed in this thesis work focuses on the capacity to compensate for persistent dynamic effects, e.g., the restoring forces and the ocean current. Indeed, since underwater movements are characterized by slow velocities and accelerations, it is appropriate to take into consideration a reduced version able to achieve null steady state error under modelling uncertainty and presence of ocean current with respect to a minimal number of parameters.

Given the gravity and buoyancy forces

$$\boldsymbol{f}_G(\boldsymbol{R}_B^I) = \boldsymbol{R}_I^B \begin{bmatrix} 0 \\ 0 \\ W \end{bmatrix} , \boldsymbol{f}_B(\boldsymbol{R}_B^I) = -\boldsymbol{R}_I^B \begin{bmatrix} 0 \\ 0 \\ B \end{bmatrix} \tag{3.12}$$

in the center of mass $\boldsymbol{r}_G^B = \begin{bmatrix} x_G & y_G & z_G \end{bmatrix}^{\mathrm{T}}$ and buoyancy $\boldsymbol{r}_B^B = \begin{bmatrix} x_B & y_B & z_B \end{bmatrix}^{\mathrm{T}}$, respectively, with W, B the weight and the buoyancy of the submerged body, the gravity-buoyancy force/moment vector in body-fixed frame Σ_B is defined

as:

$$\boldsymbol{g}(\boldsymbol{R}_B^I) = -\begin{bmatrix} \boldsymbol{f}_G(\boldsymbol{R}_B^I) + \boldsymbol{f}_B(\boldsymbol{R}_B^I) \\ \boldsymbol{r}_G^B \times \boldsymbol{f}_G(\boldsymbol{R}_B^I) + \boldsymbol{r}_B^B \times \boldsymbol{f}_B(\boldsymbol{R}_B^I) \end{bmatrix} . \tag{3.13}$$

In terms of Euler angles it is expressed as

$$\boldsymbol{g}(\boldsymbol{\eta}_2) = \begin{bmatrix} (W-B)s_\theta \\ -(W-B)c_\theta s_\phi \\ -(W-B)c_\theta c_\phi \\ -(y_G W - y_B B)c_\theta c_\phi + (z_G W - z_B B)c_\theta s_\phi \\ (z_G W - z_B B)s_\theta + (x_G W - x_B B)c_\theta c_\phi \\ -(x_G W - x_B B)c_\theta s_\phi - (y_G W - y_B B)s_\theta \end{bmatrix} \tag{3.14}$$

where it is noticeable that the linear force is constant in earth-fixed frame Σ_I and the moment of inertia are constant in the vehicle-fixed fame. Thus, $\boldsymbol{g}(\boldsymbol{\eta}_2)$ is linear with respect to the following parameters

$$\boldsymbol{\theta}_{\text{GB}} = \begin{bmatrix} -(W-B) \\ x_G W - x_B B \\ y_G W - y_B B \\ z_G W - z_B B \end{bmatrix} , \tag{3.15}$$

and it can be re-written in regressor form as

$$\boldsymbol{g}(\boldsymbol{\eta}_2) = \boldsymbol{Y}_{\text{GB}}(\boldsymbol{R}_B^I)\boldsymbol{\theta}_{\text{GB}} \tag{3.16}$$

where

$$\boldsymbol{Y}_{\text{GB}} = \begin{bmatrix} \boldsymbol{R}_I^B \boldsymbol{z} & \boldsymbol{O}_{3\times 3} \\ \boldsymbol{0}_{3\times 1} & \boldsymbol{S}(\boldsymbol{R}_I^B \boldsymbol{z}) \end{bmatrix} , \tag{3.17}$$

with $\boldsymbol{z} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^{\text{T}}$ and $\boldsymbol{S}(\cdot)$ the skew matrix performing the cross product.

The ocean current can be modelled as constant in Σ_I frame and merged with the restoring forces contribution obtaining the following regressor

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{O}_{3\times 3} & \boldsymbol{R}_I^B & \boldsymbol{O}_{3\times 3} \\ \boldsymbol{S}(\boldsymbol{R}_I^B \boldsymbol{z}) & \boldsymbol{O}_{3\times 3} & \boldsymbol{R}_I^B \end{bmatrix} \in \mathbb{R}^{6\times 9} , \tag{3.18}$$

that is the reduced regressor taking into consideration the dynamic persistent effects. Thus, the AUV dynamic model can be written as

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{Y}\boldsymbol{\theta} = \boldsymbol{\tau}_v \ , \tag{3.19}$$

which considering the steady state is

$$\boldsymbol{Y}\boldsymbol{\theta} = \boldsymbol{\tau}_v \ , \tag{3.20}$$

with $\boldsymbol{\theta}$ the dynamic parameters vector defined as

$$\boldsymbol{\theta} = \begin{bmatrix} GB_x & GB_y & GB_z & F_x & F_y & F_z & M_x & M_y & M_z \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{9\times 1} \ , \tag{3.21}$$

with

$$\begin{aligned} GB_x &= x_G W - x_B B \\ GB_y &= y_G W - y_B B \\ GB_z &= z_G W - z_B B \\ F_z &= -(W - B) \end{aligned}$$

and F_x, F_y, M_x, M_y, M_z the linear forces and moments due to external disturbances. In particular, $\boldsymbol{GB}$ is expressed in Σ_B while $\boldsymbol{F}, \boldsymbol{M}$ are expressed in Σ_I.

As known in adaptive control theory for robots [36], it is appropriate to further investigate the *identifiability* of the parameters. By SVD (Singular Value Decomposition) analysis of the regressor it has been observed that changes in roll and pitch are sufficient to excite all the parameters. On the other hand, if the vehicle is completely still, linear combinations can arise causing issues of identifiability of the related parameters. In practice small movements of the vehicle in hover are difficult to detect as *singular* case and they may cause numerical issues that can be solved resorting to the following further reduced regressor

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{O} & \boldsymbol{R}_I^B \\ \boldsymbol{S}(\boldsymbol{R}_I^B \boldsymbol{z}) & \boldsymbol{O} \end{bmatrix} \ . \tag{3.22}$$

However, the latter does not ensure to compensate for the external moment disturbances. For these reasons, the first reduced regressor in Eq. (3.18) is used taking into account the possible phenomena of bursting [5]. Indeed, the presence of issues of identifiability joint to the presence of sensor noisy measurements, as usual in practice, may cause the system to burst into an oscillation which then dies away. More in detail, the unidentifiable parameters drift according to a random walk dynamics (see Fig. 3.1 for a numerical example). Then, a small movement of the vehicle may excite the parameter under discussion and inject the model-based compensation of the drifted parameter in the control loop as a *disturbance*. Depending on the amplitude of the drift, and thus the amplitude of the disturbance, the feedback may recover it by oscillating around the desired value while correctly estimating the parameters, or it can become unstable.

The overall effect of the bursting is difficult to predict for a generic mechanical system and depends by several factors among which: the inertia, the sensor noise, the duration of the non-exciting movement, etc.

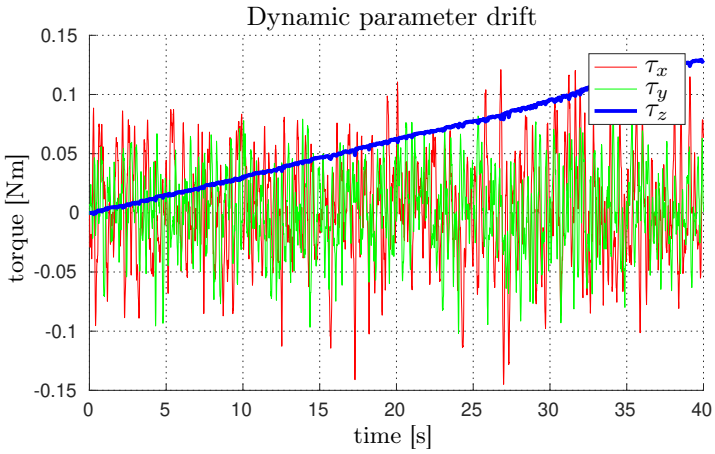

Figure 3.1: A drifting parameter example: the particular non-exciting movement and the presence of sensor noise are causing the drift of the gravity/buoyancy moment z-component.

3.1.3 Thruster Dynamics within Adaptive Control

For underwater vehicles the most common propulsion system is represented by thrusters. The latter is the lowest control level of the AUV. Differently from, e.g., industrial robotics, its dynamic effect is not straightforward since its performances are afflicted by several factors such as cross-flows and ocean current [58] and, therefore, strictly depending on its allocation [40, 37, 58]. Furthermore, most of the off-the-shelf propellers lack of any sensor to read the necessary feedback, e.g., the thrust and propeller speed rotation. However, as shown in next Sections, it is not efficient to neglect the thruster dynamics during the control design.

Thruster Dynamics

Figure 3.2 shows a scheme for a generic tunnel thruster aligned along the x direction with the main variables that characterize it. In detail, u_p is the axial flow velocity into the propeller, τ_m is the thruster motor torque, n represents the rotation shaft speed and T is the produced axial thrust. Furthermore, most of propellers are electrically driven. Thus, they present a DC-motor which receives a voltage control input v_m.

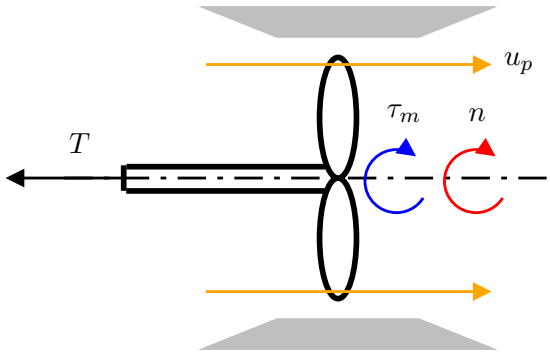

Figure 3.2: Dynamic modelling variables of the propeller

Thruster modelling is quite complex since there are several hydrodynamic factors that need to be taken into consideration. Furthermore, in real systems the measuring of all necessary quantities is not always available. For these reasons,

in literature several analysis studies have been made proposing different models. In particular, within the aim to focus the attention on the intervention case and therefore on the steady state, the model proposed by [75] has been selected.

In detail, in the latter work, using an energy-based method, the following lumped parameter model was proposed

$$\dot{n} = -\alpha n|n| + \beta v_m \tag{3.23}$$

$$T = C_t n|n| \quad , \tag{3.24}$$

where Eq. (3.23) represents the propellers dynamics with α and β constant model parameters, and Eq. (3.24) is the mapping between the propeller thrust and the rotation speed, with C_t a proportional constant that is experimentally determined.

The above model is suitable for very low velocities. However, it does not take into account any interaction with the ambient flow and thus with the eventual ocean current $\boldsymbol{\nu}_c$. Indeed, the thruster performances are differently affected by the ocean current depending on whether the vehicle is going against or in the same direction, (see Fig. 3.3). In particular, according to the considered scenarios, Eq. (3.24) is rewritten as

$$T = C_t n|n|k_1 \exp\left(k_2[0\ 1\ 0]\left(\boldsymbol{R}_{\text{th}}^{B}\right)^{\text{T}}\boldsymbol{\nu}_r\right) \tag{3.25}$$

where k_1, k_2 are further terms depending on the relative velocity $\boldsymbol{\nu}_r$ already defined in Eq. (2.19), [0 1 0] is a selection vector and $\boldsymbol{R}_{\text{th}}^{B}$ represents the thruster orientation with respect to the body-fixed frame [58]. Therefore, different effects on the produced thrust are obtained as shown in Fig. 3.4.

Force/moment-thruster mapping

As already introduced in Section 2.3.2, the mapping relation between forces / moments acting on the vehicle and the thruster control inputs $\boldsymbol{u}_{\text{tr}}$ defined in Eq. (2.26), is not linear. The relation, beyond the propeller dynamics, also

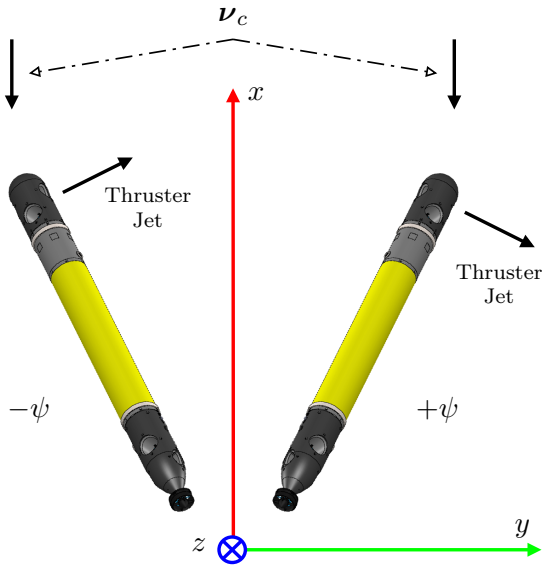

Figure 3.3: Two possible scenarios: thrusting in the same direction (negative yaw angle) or against (positive yaw angle) the ocean current.

neglects further factors which can contribute to the produced thrust degradation such as the ocean current. Thus, within the aim to consider such effects on the thrust and focusing on electrically driven propellers, Eq. (2.26) can be rewritten as [37]

$$\boldsymbol{\tau}_v = \boldsymbol{B}\boldsymbol{K}\boldsymbol{v}_m \in \mathbb{R}^6 \,, \tag{3.26}$$

where $\boldsymbol{v}_m$ is the vector collecting the voltage inputs and $\boldsymbol{K}$ is a diagonal matrix defined as

$$\boldsymbol{K} = \text{diag}\left(K_i(\boldsymbol{\nu}_r, \boldsymbol{R}_I^B, \boldsymbol{R}_{\text{th}}^B)\right) \in \mathbb{R}^{p\times p} \tag{3.27}$$

with K_i representing the i-th thruster dynamics. More in detail, each term K_i is a function of the relative velocity $\boldsymbol{\nu}_r$, necessary to take into account the ocean current effects, the vehicle orientation $\boldsymbol{R}_I^B$ and the thruster orientation $\boldsymbol{R}_{\text{th}}^B$ with respect to the body-fixed frame. It is worth noticing that the thruster orientation $\boldsymbol{R}_{\text{th}}^B$ does not represent a state variable. Hence, its presence in Eq. (3.27), as a functional dependence, is not analytically correct. However, this way results preferable to make the analysis more understandable to the reader.

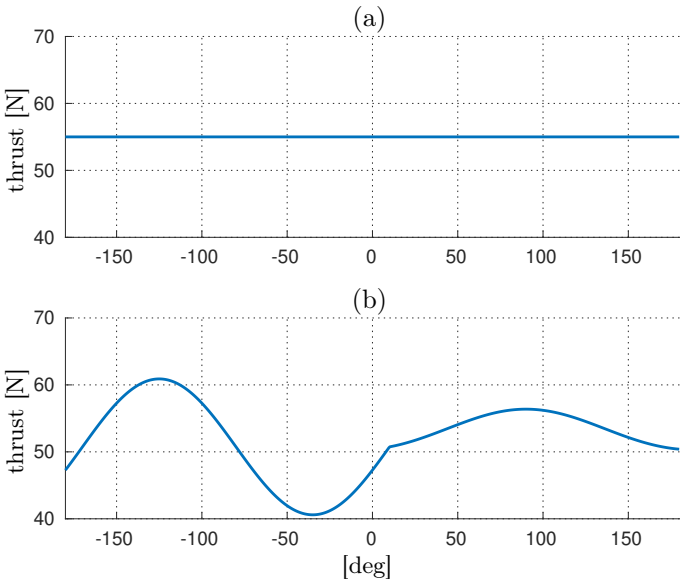

Figure 3.4: Two different thruster models are shown varying the angle formed by the thrust and ocean current directions in $[-180°; 180°]$: (a) model with very low relative velocities; (b) model for an arbitrary motion thrusting against or with the ocean current.

Thruster Dynamics inclusion

The controllers are usually designed neglecting the thruster dynamics. However, since the thruster input is usually the voltage, the designer needs to estimate the thruster gain at steady state as shown in Fig. 3.5. Let us define as $\hat{\boldsymbol{K}}_\infty \in \mathbb{R}^{p\times p}$, the diagonal matrix collecting those gains. Merging Eq. (3.26) and the controller in Eq. (3.4), the designer assumes the following relationship:

$$\boldsymbol{B}\hat{\boldsymbol{K}}_\infty \boldsymbol{v}_m = \boldsymbol{\tau}_c \, . \tag{3.28}$$

If the thrusters are all equal to each other, the matrix $\hat{\boldsymbol{K}}_\infty$ is defined as a scalar by an Identity matrix and the scalar gain is usually estimated in the lab or by trial and error for one single thruster. The voltage input is thus computed as

$$\boldsymbol{v}_m = \hat{\boldsymbol{K}}_\infty^{-1} \boldsymbol{B}^\dagger \boldsymbol{\tau}_c \, . \tag{3.29}$$

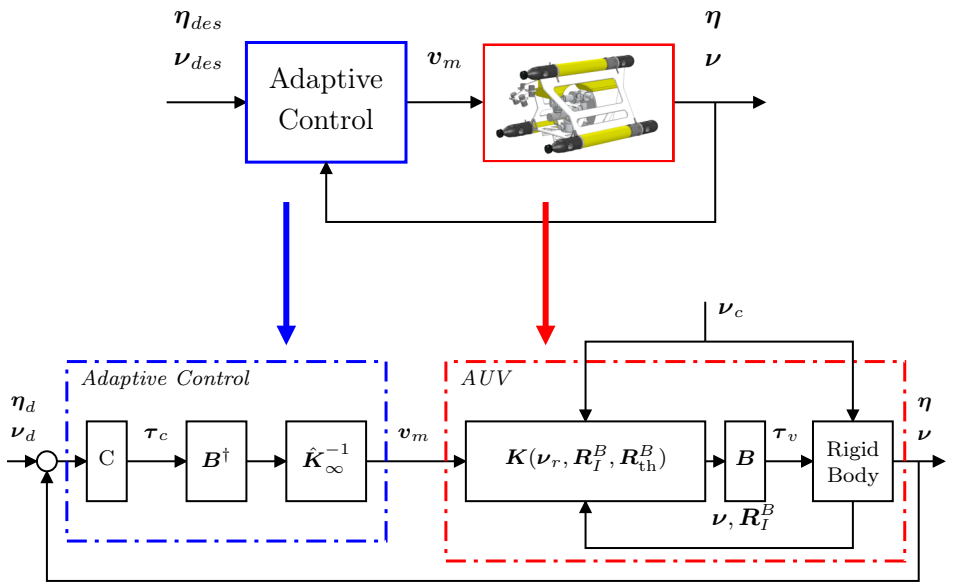

Figure 3.5: Control loop scheme: the proposed controller (in blue) and the vehicle-thruster model (in red).

However, since $\hat{\boldsymbol{K}}_\infty$ is only an approximation of $\boldsymbol{K}$ the real input to the dynamic system is

$$\boldsymbol{\tau}_v = \underbrace{\boldsymbol{B}\boldsymbol{K}\hat{\boldsymbol{K}}_\infty^{-1}\boldsymbol{B}^\dagger}_{\boldsymbol{K}_{eq}}\boldsymbol{\tau}_c \tag{3.30}$$

where, as discussed earlier

$$\boldsymbol{K} = \boldsymbol{K}(\boldsymbol{\nu}_r, \boldsymbol{R}_i^B, \boldsymbol{R}_{th}^B)\ , \tag{3.31}$$

i.e., each thruster *reacts* in a different manner depending on its location on the vehicle with respect to the presence of the ocean current and the eventual error in the steady state estimate. Thus, there exists a *distortion* effect on the rigid body dynamics.

The mapping of the control action $\boldsymbol{\tau}_c$ to the physical wrench acting on the vehicle $\boldsymbol{\tau}_v$ is distorted by the (6×6) matrix:

$$\boldsymbol{K}_{eq}(\boldsymbol{\nu}_r, \boldsymbol{R}_i^B, \boldsymbol{R}_{th}) = \boldsymbol{B}\boldsymbol{K}\hat{\boldsymbol{K}}_\infty^{-1}\boldsymbol{B}^\dagger \tag{3.32}$$

which is assumed to be Identity by the designer while it is, on the contrary, constant at steady state and not even diagonal and couples the control directions.

With the risk to be redundant let us recopy eq. (3.4) merged with the relations above:

$$\boldsymbol{\tau}_v = \boldsymbol{K}_{eq}\boldsymbol{Y}\hat{\boldsymbol{\theta}} + \boldsymbol{K}_{eq}\boldsymbol{K}_s\boldsymbol{s} \tag{3.33}$$

which clearly shows two *distortion* effects, first on the identified parameters and then on the control gains. The latter has potentially a dramatic effect, i.e., that the effective control gains are different from the ones tuned by trail-and-error in the pool due to the different working conditions.

3.2 UVMS Adaptive Control

The Adaptive Control law can be applied to Underwater Vehicle Manipulator Systems as well [7]. In particular, the vehicle control is designed within the aim to compensate for the dynamic interactions due to the manipulator presence resorting to an adaptive, recursive approach. Then, in the following, the system dynamic model define in Eq. (2.22) is rewritten in such a way to remark the interaction between consecutive rigid bodies of the serial chain.

Considering the convention according to which the superscript is the reference frame Σ_i, that is dropped for quantities referring to the inertial frame, let consider the velocity propagation along the chain

$$\boldsymbol{\nu}_{i+1}^{i+1} = \boldsymbol{U}_{i+1}^{i^{\mathrm{T}}}\boldsymbol{\nu}_i^i + \dot{q}_{i+1}\boldsymbol{z}_i^{i+1} \quad i = 0, \ldots, n-1 \tag{3.34}$$

where:

$$\boldsymbol{z}_i^i = \begin{cases} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^{\mathrm{T}} & \text{if rotational joint} \\ \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}} & \text{if prismatic joint} \end{cases}$$

and the matrix $\boldsymbol{U}_{i+1}^{i} \in \mathbb{R}^{6\times6}$ is defined as

$$\boldsymbol{U}_{i+1}^{i} = \begin{bmatrix} \boldsymbol{R}_{i+1}^{i} & \boldsymbol{O}_{3\times3} \\ \boldsymbol{S}(\boldsymbol{r}_{i,i+1}^{i})\boldsymbol{R}_{i+1}^{i} & \boldsymbol{R}_{i+1}^{i} \end{bmatrix} \tag{3.35}$$

with $\boldsymbol{R}_{i+1}$ the matrix expressing the rotation from frame Σ_{i+1} to Σ_i, $\boldsymbol{S}(\cdot)$ the skew matrix operator and $\boldsymbol{r}_{i,i+1}$ the vector pointing from Σ_i to Σ_{i+1} expressed with respect to Σ_i. Thus, the total generalized force vector acting on the i-th body is

$$\boldsymbol{h}_{t,i}^{i} = \boldsymbol{h}_{i}^{i} - \boldsymbol{U}_{i+1}^{i}\boldsymbol{h}_{i+1}^{i+1} \quad i = 1, \dots, n-1 \tag{3.36}$$

$$\boldsymbol{h}_{t,n}^{n} = \boldsymbol{h}_{n}^{n} \,, \tag{3.37}$$

where $\boldsymbol{h}_{i}^{i}$ is the generalized force actuated by body $i-1$ on body i. Therefore, $\boldsymbol{h}_{0}^{0} = \boldsymbol{\tau}_v$ since the vehicle is the first body of the chain. Then, Eq. (2.18) can be rewritten in body frame Σ_i^i as

$$\boldsymbol{M}_i\dot{\boldsymbol{\nu}}_i^i + \boldsymbol{C}_i(\boldsymbol{\nu}_i^i)\boldsymbol{\nu}_i^i + \boldsymbol{D}_i(\boldsymbol{\nu}_i^i)\boldsymbol{\nu}_i^i + \boldsymbol{g}_i^i(\boldsymbol{R}_i) = \boldsymbol{h}_{t,i}^i \tag{3.38}$$

that in regressor form is

$$\boldsymbol{Y}(\boldsymbol{R}_i, \boldsymbol{\nu}_i^i, \dot{\boldsymbol{\nu}}_i^i)\boldsymbol{\theta}_i = \boldsymbol{h}_{t,i}^i \,. \tag{3.39}$$

Within the aim to compensate for the vehicle while taking into account the manipulator dynamic effects, the regressor can be achieved through an iterative Newton-Euler-based approach. In detail, merging Eq. (3.36) and Eq. (3.38) yields

$$\boldsymbol{\tau}_v = \boldsymbol{h}_0^0 = \boldsymbol{Y}_0\theta_0 + \boldsymbol{U}_1^0\boldsymbol{h}_1^1 \tag{3.40}$$

where $\boldsymbol{Y}_0, \boldsymbol{\theta}_0$ are the regressor and dynamic parameters vector, respectively, of the corresponding rigid body (body 0 that is the vehicle in the specific case). The latter relation can be rewritten substituting the force $\boldsymbol{h}_1^1$ for the first link as

$$\boldsymbol{h}_0^0 = \boldsymbol{Y}_0\boldsymbol{\theta}_0 + \boldsymbol{U}_1^0\big(\boldsymbol{Y}_1\boldsymbol{\theta}_1 + \boldsymbol{U}_2^1\boldsymbol{h}_2^2\big) \tag{3.41}$$

$$= \boldsymbol{Y}_0\boldsymbol{\theta}_0 + \boldsymbol{U}_1^0\boldsymbol{Y}_1\boldsymbol{\theta}_1 + \boldsymbol{U}_1^0\boldsymbol{U}_2^1\boldsymbol{h}_2^2 \,. \tag{3.42}$$

Iterating the operation and rearranging the terms yields

$$\boldsymbol{h}_0^0 = \underbrace{\begin{bmatrix} \boldsymbol{Y}_0 & \boldsymbol{U}_1^0\boldsymbol{Y}_1 & \cdots & \boldsymbol{U}_1^0\boldsymbol{U}_2^1 & \cdots & \boldsymbol{U}_n^{n-1}\boldsymbol{Y}_n \end{bmatrix}}_{\overline{\boldsymbol{Y}}} \underbrace{\begin{bmatrix} \boldsymbol{\theta}_0 \\ \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \\ \vdots \\ \boldsymbol{\theta}_n \end{bmatrix}}_{\overline{\boldsymbol{\theta}}} \tag{3.43}$$

that is computationally effective. Thus, the adaptive control law defined for the AUV in Eq. (3.4) is now defined for a UVMS as

$$\boldsymbol{\tau}_c = \overline{\boldsymbol{Y}}\,\hat{\overline{\boldsymbol{\theta}}} + \boldsymbol{K}_s\boldsymbol{s} \tag{3.44}$$

with the update law

$$\dot{\hat{\overline{\boldsymbol{\theta}}}} = \boldsymbol{K}_\theta^{-1}\overline{\boldsymbol{Y}}^{\mathrm{T}}\boldsymbol{s} \tag{3.45}$$

where $\boldsymbol{K}_s \in \mathbb{R}^{6\times 6} > \boldsymbol{O}$ and $\boldsymbol{K}_\theta \in \mathbb{R}^{\mu\times\mu} > \boldsymbol{O}$.

The stability analysis for this control law is not presented since it can be performed in a way similar to the one reported in Section 3.1.1.

3.2.1 UVMS Reduced controller

Aimed at compensating for persistent dynamic effects guaranteeing the null steady state error and in order to avoid the joint accelerations measurement, since off-the-shelf manipulators are usually equipped only with low level position-velocity control sensors, a reduced version of the controller is considered, as done for the AUV. In particular, for the vehicle (body 0) the regressor already defined in Eq. (3.18) can be used. For the manipulator, however, the regressor in Eq. (3.22) has to be adopted, otherwise identifiability issues in linear combination could arise in building the regressor $\overline{\boldsymbol{Y}}$. Therefore, the total UVMS regressor is $\overline{\boldsymbol{Y}} \in \mathbb{R}^{6\times 9+6n}$.

3.3 Adaptive Control Simulations

In the following the numerical validations of the adaptive control law are reported.

3.3.1 AUV simulations

Numerical validations are performed taking into consideration the dynamic model of the underwater vehicle developed within the European Project ROBUST, that is well described further on. In particular, several case studies are presented doing a comparison with a PID controller.

In order to show the goodness of the adaptive control, the ocean current $\boldsymbol{\nu}_c = \begin{bmatrix} 0.2 & 0.2 & 0 & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$ in [m/s] is considered. Furthermore, a preliminary learning phase is performed to let the adaptive controller identify the dynamic parameters, e.g., the force necessary to compensate $\boldsymbol{\nu}_c$. Indeed, thanks to its parameters update law, which is based on the system state error, the controller is able to identify the force components necessary for compensating the ocean current disturbance with no need of sensors or particular navigation filters.

Figure 3.6 shows the force components identified during the learning phase. It is noticeable that F_z is null since the vehicle has a neutral buoyancy. The identified values are used in the next simulations to initialize the corresponding components inside the dynamic parameters vector $\boldsymbol{\theta}$.

All simulations are performed with a sampling time $T_s = 10$ ms and a trapezoidal velocity profile is considered for the vehicle required movements.

The *first case study* considers a station keeping task. It allows to show the effectiveness of the learning phase performed with the adaptive controller. Indeed, as shown in Fig. 3.7, the position error norm of the adaptive control is null whereas the PID has a non-null error due to its transient necessary to compensate for the ocean current disturbance.

In the *second case study* a movement varying the pitch angle from 0 to $\frac{\pi}{12}$ rad and then back to 0 is considered. As noticeable from Fig. 3.8, the PID error is

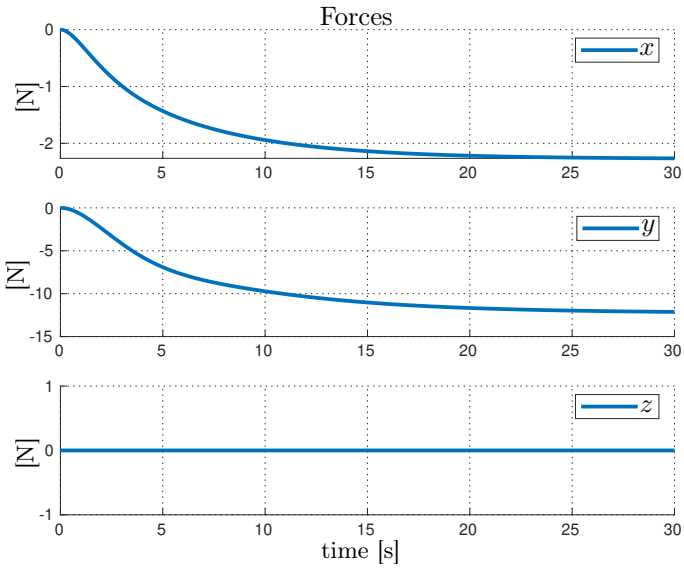

Figure 3.6: Adaptive learning phase: dynamic parameters corresponding to force components due to the ocean current compensation.

much higher than adaptive one. Indeed, the PID controllers performs in Σ_B, therefore, during the transient the integral action acts as disturbance on z-axis and pitch moment in body frame until its discharge. On the other hand the adaptive controller compensates for the dynamic effect in the proper frame and thus considers the external disturbances in Σ_I (see Forces and Moments plots in Fig. 3.9) and the $\boldsymbol{GB}$ moments in Σ_B (see $\boldsymbol{GB}$ plot in Fig. 3.9).

In the *third case study* a movement varying the yaw angle from 0 to $\frac{\pi}{2}$ rad and then back to 0 is simulated. As in the previous case, the PID controller has the drawback of waiting the integral discharge before to compensate in the right direction. Indeed, the integral action compensating for the ocean current in body frame are switched when the vehicle rotates with yaw $\psi = \frac{\pi}{2}$, acting as disturbance during the transient (see Fig. 3.10) whereas the adaptive controller compensates considering $\boldsymbol{\nu}_c$ in the earth-fixed frame (see Fig. 3.11) and therefore avoiding the PID transient drawback.

The *fourth case study* consists of a path-following on the xy plane with a final descent (see Fig. 3.12). The corresponding position and orientation error

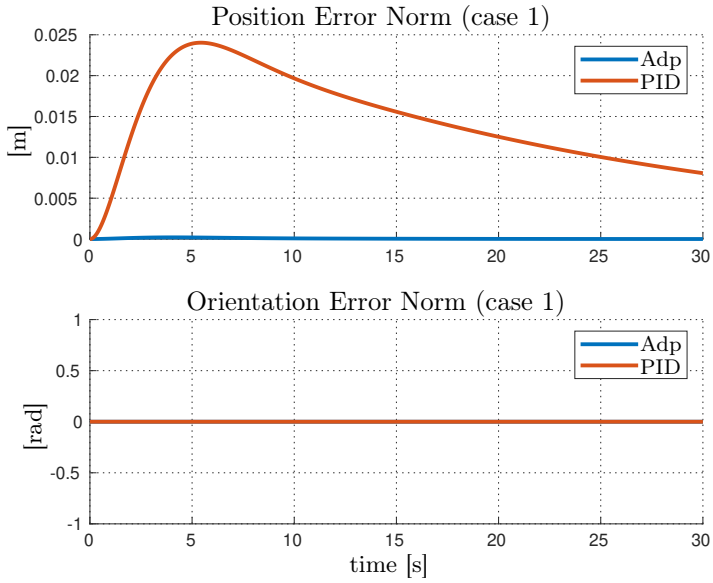

Figure 3.7: Case 1 - Station keeping simulation: comparison between adaptive and PID error norms.

norm are reported in Fig. 3.13 while Fig. 3.14 shows the dynamic parameters identified by the adaptive controller.

AUV simulations with Thruster Dynamics inclusion

An open frame vehicle is considered for the case study whose dynamic parameters have been identified in [17]. The actuators, however, have been *changed* according to the thruster configuration shown in Fig. 2.5 with the TCM defined in Eq. (2.28). For the single thruster model, the parameters which have been experimentally determined in [75], are used.

An intervention operation and thus a station-keeping task is considered as case study. In particular, the vehicle is required to perform a small movement, with a trapezoidal velocity profile, along the x-direction maintaining the own orientation. More in detail, the case study is simulated with a sampling time $T_s = 10$ ms in three different conditions: the ideal condition where the thruster dynamics is neglected, the real one where the propeller dynamics with thruster

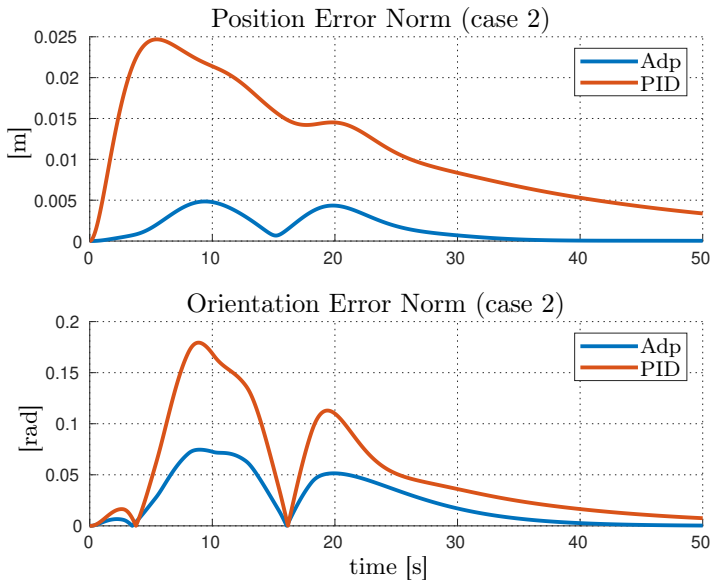

Figure 3.8: Case 2 - Pitch movement simulation: comparison between adaptive and PID error norms.

degradation is considered and the latter with the presence of the ocean current which increases the thrust degradation. Indeed, in the most of cases the gain tuning phase is performed in a water tank, therefore with no ocean current, and, as mentioned above, neglecting the actuator dynamics due to the lack of sensors. Then, the resulting values are also used offshore. However, the controller performance deterioration has to be taken into account since the different operative conditions and the presence of propellers change the plant used to design the controller resulting in a $\boldsymbol{K}_{eq}$ different from the identity matrix.

Figures 3.15 and 3.16 show the pose error and vehicle force norm, respectively. In particular, it is noticeable how the error increases from the Ideal condition to the Real one with the presence of the ocean current. However, the controller results stable with null steady state error.

Figure 3.17 represents the dynamic parameters corresponding to the linear components for compensating the external disturbances and the gravity-buoyancy effect. In detail, it is observable that the presence of the ocean current $\boldsymbol{\nu}_c =$

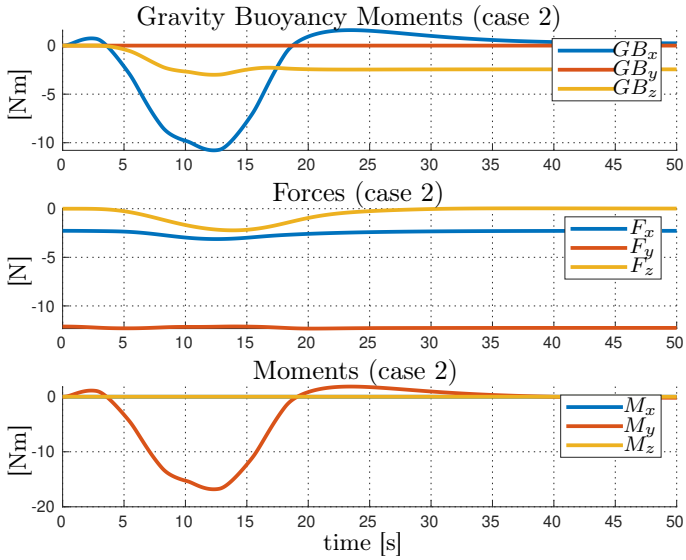

Figure 3.9: Case 2 - Pitch movement simulation: adaptive controller dynamic parameters.

$[.2\ .2\ 0\ 0\ 0\ 0]^{\mathrm{T}}$ influences the x, y components differently from the other two conditions. Furthermore, the thruster dynamics included in $\boldsymbol{K}_{eq}$ makes a clear distortion on the gravity-buoyancy component (z-component) which looses in this way its physical significance. To better understand, the $\boldsymbol{K}_{eq}$ corresponding to the last sample of the Real condition simulations without and with the ocean current, respectively, are reported:

$$\begin{bmatrix} -.0755 & 0 & 0 & 0 & 0 & 0 \\ 0 & -.0755 & 0 & 0 & 0 & .0002 \\ 0 & 0 & .5015 & 0 & 0 & 0 \\ 0 & 0 & 0 & .5015 & 0 & 0 \\ 0 & 0 & 0 & 0 & .5015 & 0 \\ 0 & 0 & 0 & 0 & 0 & -.0755 \end{bmatrix}$$

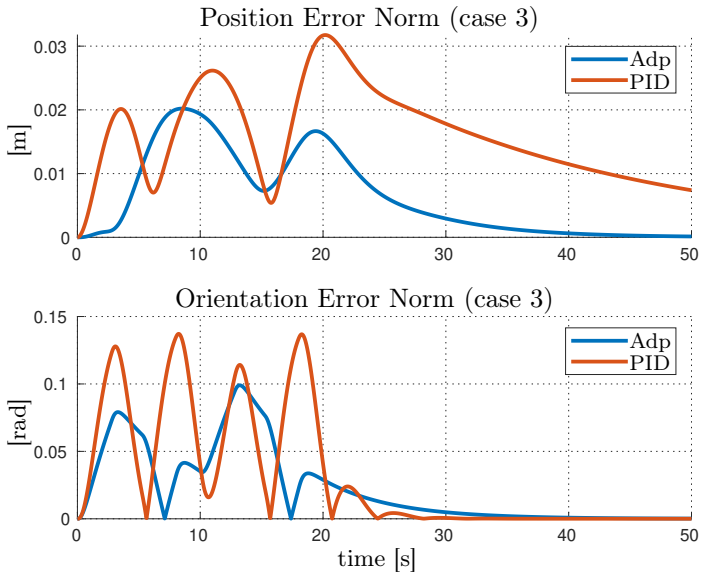

Figure 3.10: Case 3 - Yaw movement simulation: comparison between adaptive and PID error norms.

and

$$\begin{bmatrix} .4756 & -.0068 & 0 & 0 & 0 & .0269 \\ -.0068 & .4756 & 0 & 0 & 0 & -.0269 \\ 0 & 0 & .5015 & 0 & 0 & 0 \\ 0 & 0 & 0 & .5015 & 0 & 0 \\ 0 & 0 & 0 & 0 & .5015 & 0 \\ .0067 & -.0067 & 0 & 0 & 0 & .4756 \end{bmatrix}.$$

From these values it is clear that, differently from the Ideal case where $\boldsymbol{K}_{eq}$ corresponds to the Identity matrix, in the Real condition the diagonal elements (in blue) can vary according to the specific system and movement and furthermore there may be out-diagonal elements (in red) corresponding to direction couplings that increase with the ocean current.

To further underline the effects of the $\boldsymbol{K}_{eq}$ matrix, Fig. 3.18 shows a comparison between the dynamic parameters identified in the previous simulation (case (a)) and the ones obtaining with a 50% overestimation of $\hat{\boldsymbol{K}}_{\infty}$ (case (b)). Thus, on equal terms, the gain estimate also has a non-negligible effect.

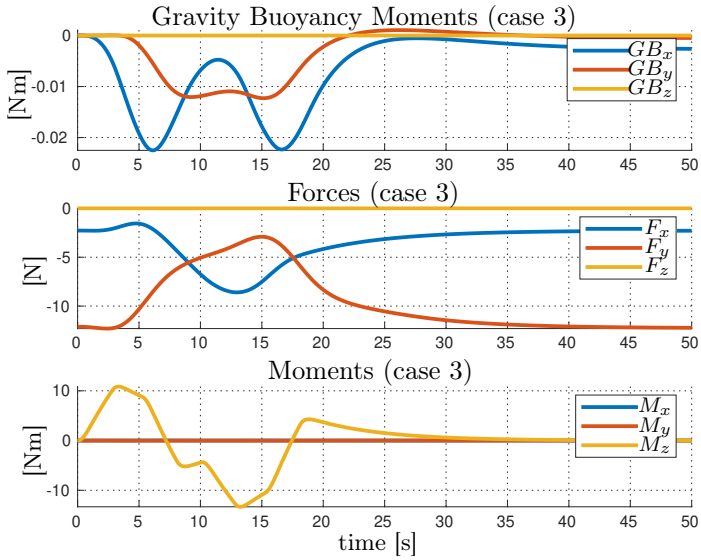

Figure 3.11: Case 3 - Yaw movement simulation: adaptive controller dynamic parameters.

Another consideration finally comes out from numerical simulations. In particular, the $\boldsymbol{K}_{eq}$ matrix presents some off-diagonal elements that are always null. This means that some control directions can not be coupled by construction and they depend on the specific TCM. In this work, given the chosen thruster allocation, the following results:

$$\boldsymbol{K}_{eq} = \begin{bmatrix} * & * & 0 & 0 & 0 & * \\ * & * & 0 & 0 & 0 & * \\ 0 & 0 & * & * & * & 0 \\ 0 & 0 & * & * & * & 0 \\ 0 & 0 & * & * & * & 0 \\ * & * & 0 & 0 & 0 & * \end{bmatrix}, \tag{3.46}$$

where the $*$ symbol represents the generic diagonal element which varies around the value 1 and the generic out-diagonal element that varies around the value 0.

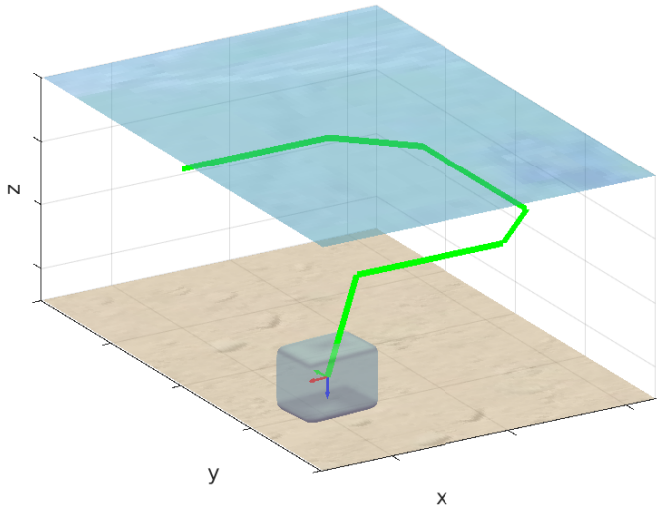

Figure 3.12: Case 4 - Path following simulation: view of the path followed by the vehicle.

3.3.2 UVMS simulations

Aimed at showing the effectiveness of the adaptive control compensating for dynamic interaction due to the manipulator movements, the dynamic model identified in [17] is used (as for the thruster inclusion simulations) along with the model of the 7 DOFs arm used in the TRIDENT project [68]. In this configuration, the arm represents the 6% of the total dry weight.

The case study consists of a trajectory tracking given in terms of end-effector position and orientation with a trapezoidal velocity profile. Furthermore, the trajectory is generated through the task priority algorithm taking into consideration as secondary task the vehicle null roll-pitch and the manipulator manipulability. The desired position/orientation along with the corresponding velocities are sent as reference to the adaptive controller and to the arm joint low-level controller, a PID in the specific case. The adaptive reduced control is used taking into consideration the regressor defined in Eq. (3.18) for the vehicle and in Eq. (3.22) for each arm link. Therefore, 9 parameters for the vehicle and $7 \cdot 6 = 42$ parameters for the arm are considered.

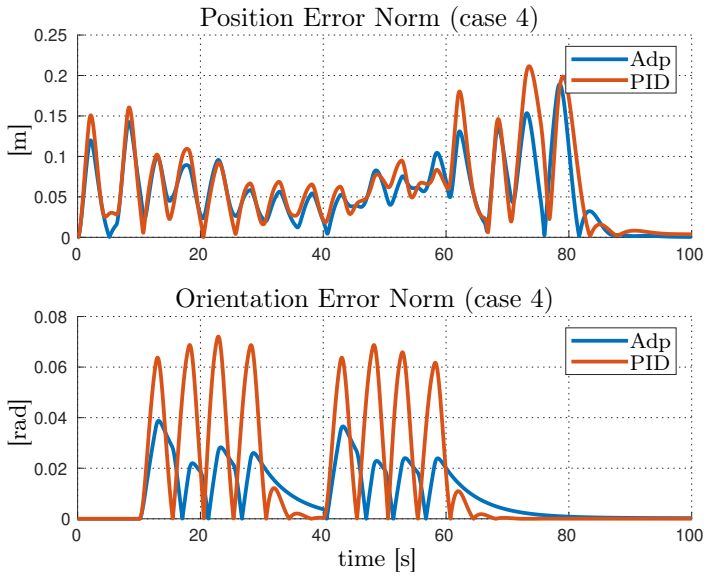

Figure 3.13: Case 4 - Path following simulation:comparison between adaptive and PID error norms.

Figure 3.19 reports the computed control forces, showing a difference in the control moments whereas the control forces are approximately equal. The controller with the arm knowledge compensates for the manipulator movement to keep the vehicle null pitch-yaw and thus its control moment is higher. Figure 3.20 and 3.21 show the vehicle and end-effector position/orientation error norms. In particular, the end-effector orientation error is expressed in quaternion convention. As noticeable, the controller with the arm knowledge has better performances. Focusing on the vehicle orientation error norm, it is observable that the adaptive control with arm knowledge is 20% lower than the peak error obtained by the other controller. Therefore, even if the arm is 6% of the total system dry weight, the proposed control goodness is effective.

3.3.3 Conclusions

The simulations results have shown the goodness of the adaptive control which, after a preliminary learning phase, is able to compensate for the gravity/buoyancy

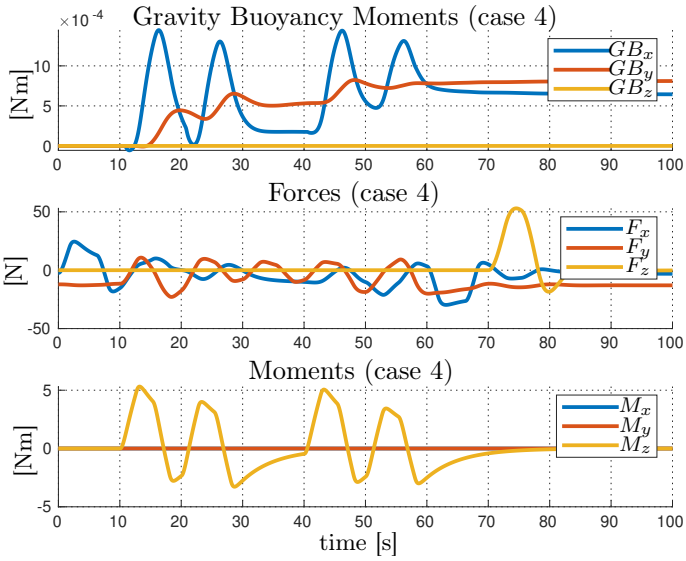

Figure 3.14: Case 4 - Path following simulation: adaptive controller dynamic parameters.

effects and the ocean current disturbance in the proper frame with no drawbacks during the transient time.

The full-dimensional adaptive control of an underwater vehicle, including the thruster dynamics and the ocean current effects, has been investigated as well. The performed analysis has shown that these effects cause the distortion of the vehicle control input $\boldsymbol{\tau}_v$ and they make the dynamics parameters loose their physical meaning. However the thruster dynamics inclusion is usually not possible in real scenarios due to the lack of any feedback sensors in most of the off-the-shelf propellers.

The effectiveness of an adaptive recursive control able to counteract for the dynamic interaction caused by the arm has been also shown.

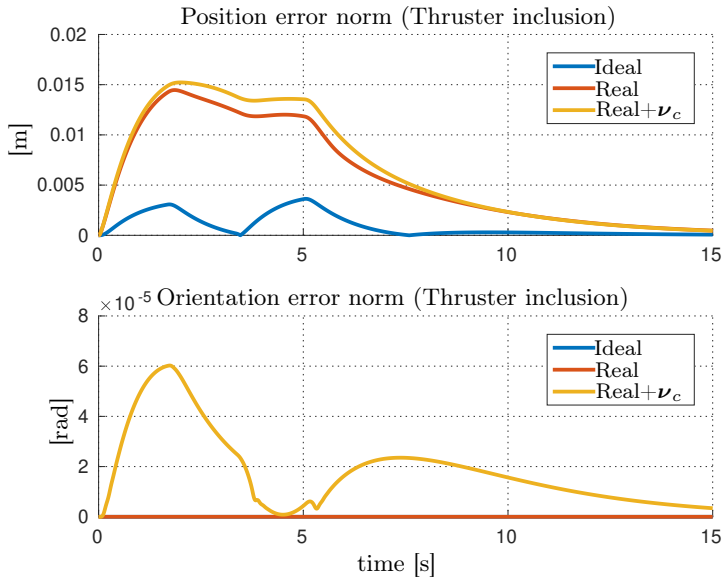

Figure 3.15: Position and orientation error norm plots: in blue the Ideal condition (no thruster dynamics); in red the Real condition (with thruster dynamics); in yellow the Real one with the presence of the ocean current.

3.4 The ROBUST project

Robotic subsea exploration technologies - ROBUST [3] is a research project funded from the European Union's Horizon 2020 research and innovation programme. The system developed within this framework (see Fig. 3.22) has been conceived for performing sea bed material identification, manganese nodules in the specific case, merging the capabilities of an Autonomous Underwater Vehicle (AUV) and a robotic manipulator equipped with a Laser Induced Breakdown Spectroscopy (LIBS) sensor.

In particular, the envisioned mission of the ROBUST UVMS consists of four steps:

- The UVMS initially performs a survey of the working area at a high altitude (about 50 m) within the aim to collect Multi Beam Echo Sounder (MBES) data.

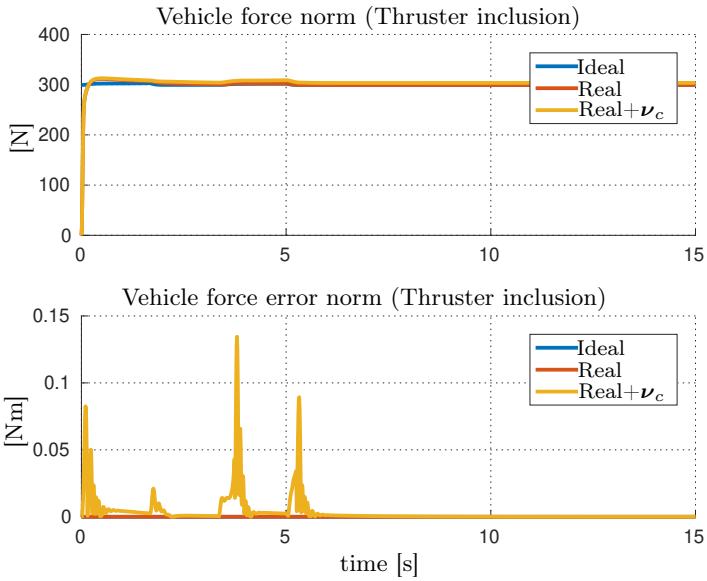

Figure 3.16: Vehicle force and moment norm plots: in blue the Ideal condition (no thruster dynamics); in red the Real condition (with thruster dynamics); in yellow the Real one with the presence of the ocean current.

- The MBES data are automatically processed for creating bathymetric maps of the area which are then used by the UVMS to autonomously determine the subarea with the highest probability of manganese nodules presence.
- The UVMS performs a low-altitude survey of the subarea to identify possible nodules in real-time using a color camera and a detection algorithm based on a Convolutional Neural Network (CNN).
- When the detection algorithm identifies a possible nodule, the UVMS lands on the seabed and the manipulator performs an analysis of the mineral through the LIBS. This step is repeated until enough data are acquired within the area of interest.

For the UVMS implementation a modular approach has been adopted. Indeed, as noticeable from Fig. 3.22, the ROBUST Underwater Vehicle Manipulator System (UVMS) consists of a rigid frame which connects three torpedo-shaped

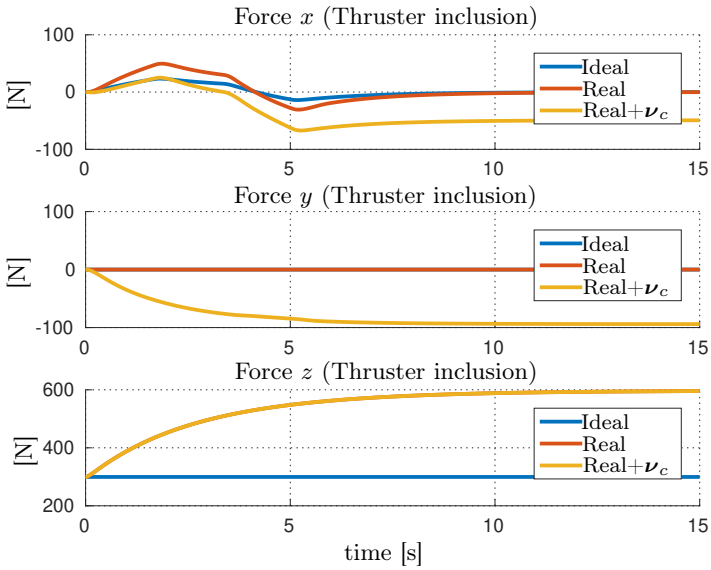

Figure 3.17: Dynamic parameters corresponding to linear components: in blue the Ideal condition (no thruster dynamics); in red the Real condition (with thruster dynamics); in yellow the Real one with the presence of the ocean current.

AUVs and a 7 DOFs manipulator manufactured by GraalTech, an Italian company that is one of the project partners. The overall system architecture is shown in Fig. 3.23. From the latter it is noticeable how the control architecture is based on the cascade of three main blocks:

1. The Mission Control Module is in charge of supervising the execution of the current *mission*, and generates the corresponding *actions* to be executed by the Kinematic Control Layer.

2. The Kinematic Control Layer (KCL) is in charge of reactively accomplishing the *control objectives* that make up the current action to be executed, generating the desired system velocity vector.

3. The Dynamic Control Layer (DCL) tracks the desired system velocity vector by generating appropriate force/torques commands for the vehicle and the manipulator.

The actions sent by the Mission Control are lists of prioritized control objectives

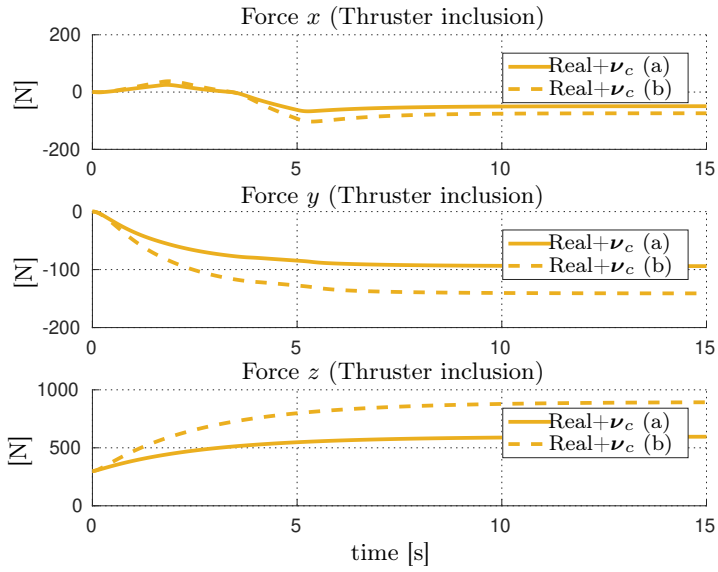

Figure 3.18: Dynamic parameters corresponding to linear components: in solid line the Real condition with the exact $\hat{\boldsymbol{K}}_\infty$; in dashed line the Real condition with a 50% overestimation of $\hat{\boldsymbol{K}}_\infty$.

which allow to perform the different phases of the ROBUST mission. More in detail, the following actions are considered:

- $\mathcal{A}_a$: UVMS heading alignment to a target position, taking into consideration some safety-related tasks, i.e., the minimum altitude and the horizontal attitude task.
- $\mathcal{A}_g$: UVMS movement towards a desired goal position, aligning the heading to the direction of the generated linear velocity while considering the safety-related tasks as defined for the previous action.
- $\mathcal{A}_p$: UVMS positioning w.r.t. the nodule, performing the system distance and the laser frame longitudinal alignment to the nodule tasks while taking into account the safety-related control objectives.
- $\mathcal{A}_d$: UVMS landing, performing, in addition to nodule distance and vehicle longitudinal alignment tasks, also an altitude task, simultaneously fulfilling the safety control objectives except the minimum altitude since the goal is to land on the seabed.

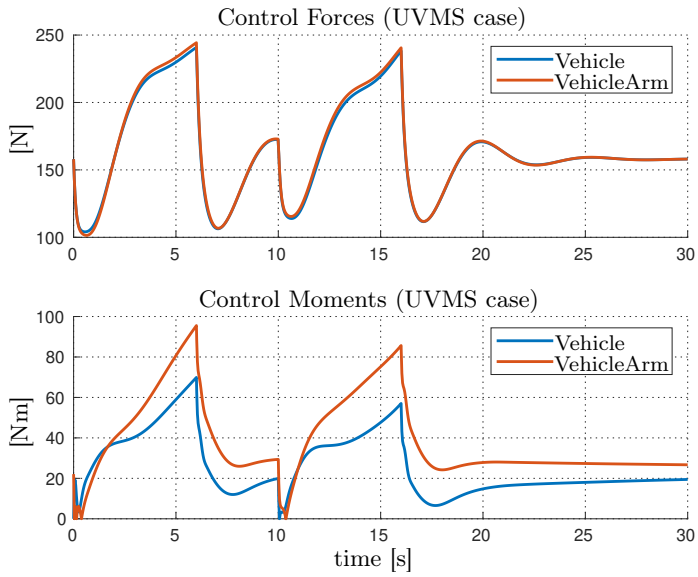

Figure 3.19: UVMS simulation: comparison between the control forces/moments by applying the adaptive control to the vehicle with (in red) and without (in blue) the arm knowledge.

Aimed at sequencing the above mentioned actions, a control state machine has been implemented, as shown in Fig. 3.24. In particular, when the system is initialized, it is set to the HOLD state, meaning that the current vehicle position is kept. When the Mission Control Module sends a specific action then the system switches to the state with the control objectives corresponding to the received action. It reverts to the HOLD state only when the required action tasks are fulfilled.

Concerning the dynamic control layer, a Proportional-Integral (PI) controller along with the adaptive control law proposed in Section 3.1 have been implemented with the possibility to switch between them at run time. However, since the framework dynamic layer receives only the velocity reference the adaptive controller has been modified removing the pose reference from the error variable $\boldsymbol{s}$ defined in Eq. (3.3) obtaining, therefore, only the velocity error:

$$\boldsymbol{s} = \boldsymbol{\nu}_d - \boldsymbol{\nu} \ . \tag{3.47}$$

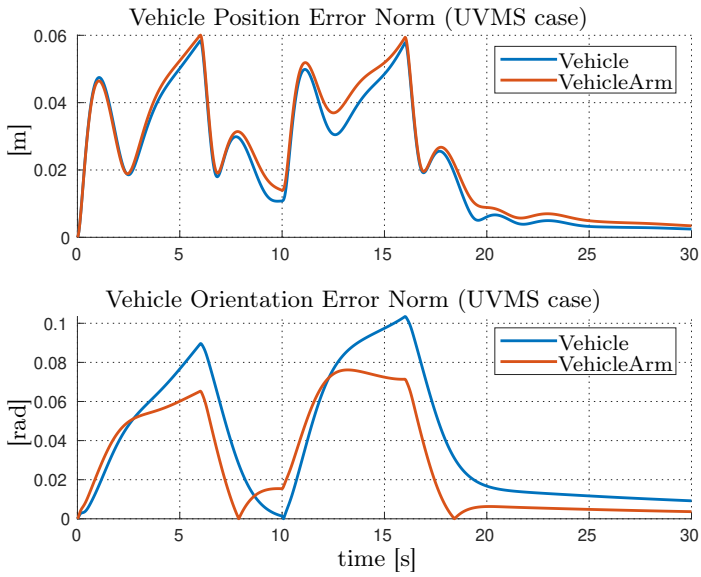

Figure 3.20: UVMS simulation: comparison between the vehicle position/orientation error norms by applying the adaptive control to the vehicle with (in red) and without (in blue) the arm knowledge.

3.4.1 Preliminary Experiments

Preliminary sea experiments have been conducted in La Spezia, at the base of the Italian Navy. In particular, focusing on the dynamic control layer, some tests performing the $\mathcal{A}_g$ action, that is movement towards desired goal positions, have been done under the effect of the ocean current.

More in detail, the following waypoints on the $xy-$plane with respect to the NED frame system have been considered:

$$
\begin{aligned}
G_1 &\rightarrow x = -8, \quad y = 21, \\
G_2 &\rightarrow x = -28, \; y = 28, \\
G_3 &\rightarrow x = -35, \; y = 8, \\
G_4 &\rightarrow x = -15, \; y = 1.
\end{aligned}
$$

Furthermore, within the aim to make a comparison, experiments have been performed both with the adaptive and PI control. The following gains have

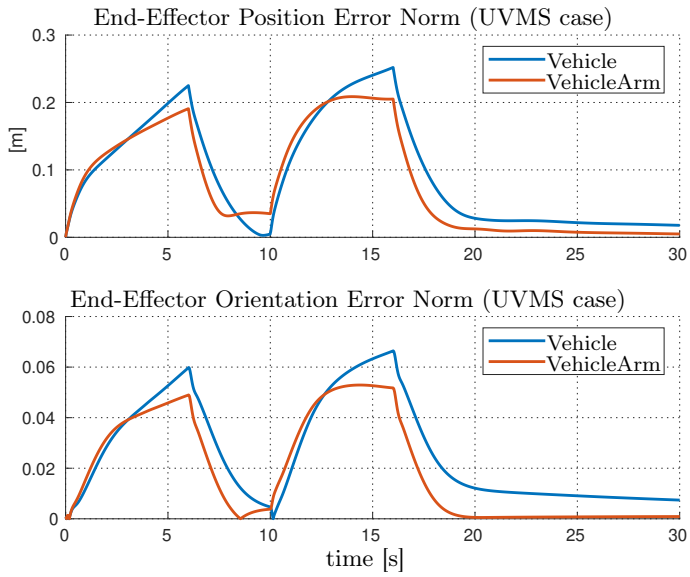

Figure 3.21: UVMS simulation: comparison between the end-effector position/orientation error norms by applying the adaptive control to the vehicle with (in red) and without (in blue) the arm knowledge.

been used for the adaptive control:

$$\boldsymbol{K}_s = \text{diag}\Big(\begin{bmatrix} 333.33 & 416 & 555 & 350 & 175 & 176 \end{bmatrix} \Big) \in \mathbb{R}^{6\times 6}$$
$$\boldsymbol{K}_\theta = \text{diag}\Big(\begin{bmatrix} 20 & 20 & 20 & 40 & 40 & 20 & 20 & 20 & 50 \end{bmatrix} \Big) \in \mathbb{R}^{9\times 9} ,$$

whereas the following ones have been used for the PI control:

$$\boldsymbol{K}_P = \text{diag}\Big(\begin{bmatrix} 333.33 & 416 & 555 & 350 & 175 & 176 \end{bmatrix} \Big) \in \mathbb{R}^{6\times 6}$$
$$\boldsymbol{K}_I = \text{diag}\Big(\begin{bmatrix} 83 & 208 & 554 & 0 & 0 & 152 \end{bmatrix} \Big) \in \mathbb{R}^{6\times 6} .$$

Figure 3.25 reports the waypoint navigation obtained running both the controllers with the corresponding control forces and moments shown in Fig. 3.26 and Fig. 3.27, respectively. From the plots, it is noticeable that the PI presents a larger error when the UVMS changes its heading in proximity of the desired goal positions. This difference of performance is observable also in the control moments whereas the control linear effort is approximately the same. Thus, the

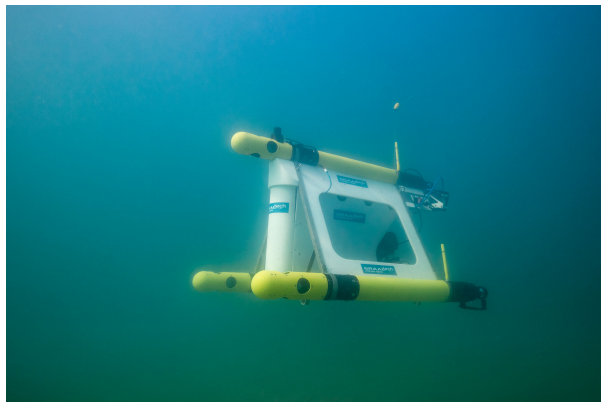

Figure 3.22: UVMS developed within the EU funded ROBUST Project.

adaptive control has a better performance during the vehicle heading change since it properly compensates for the effect of the ocean current.

Figure 3.28, 3.29 present the linear and angular velocity error norms, respectively. As observed above, the linear components have almost the same trend whereas the angular components present a slightly different behaviour. Finally Fig. 3.30 shows the dynamic parameters identified by the adaptive control. It is worth noticing that, since the vehicle movement is performed using the distance error with respect to the desired goal position without imposing a trajectory, the parameter identification is highly influenced by the integral action. Then, the dynamic parameters actually loose their physical meaning during the transient.

As further confirm of the slightly different performances between the two controllers, the integral of the velocity error $\boldsymbol{s}$ as defined in Eq. (3.47) has been computed according to the following relation:

$$e = \frac{1}{T_f} \int_0^{T_f} \|\boldsymbol{s}\| dt \tag{3.48}$$

As noticeable, the error integral is normalized with respect to the final time T_f. Since the two experiments have different time lengths, the adaptive final time (the shortest) has been chosen. Thus, the following error integrals have been

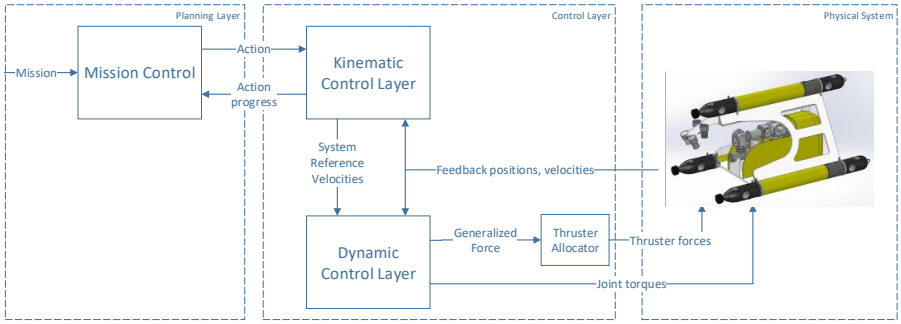

Figure 3.23: ROBUST system overall architecture: the Kinematic Control Layer implements a task priority based approach, executing the current action scheduled by the Mission Control Module; the output of the Kinematic Control Layer are the system velocities, tracked by the underlying Dynamic Control Layer.

obtained for the linear components

$$e_{\mathrm{Adp},l} = 0.28, \qquad e_{\mathrm{PI},l} = 0.30$$

and the angular ones

$$e_{\mathrm{Adp},a} = 0.05, \qquad e_{\mathrm{PI},a} = 0.07\ .$$

Therefore, these results confirm the information noticeable from the above plots. In particular,

$$e_{\mathrm{PI},l} = k_l \cdot e_{\mathrm{Adp},l}$$
$$e_{\mathrm{PI},a} = k_a \cdot e_{\mathrm{Adp},a}$$

with $k_l = 1.07$ and $k_a = 1.4$. Then, the adaptive control actually presents a lower error thanks to its parameter adaption in the proper reference frame whereas the PI control, compensating for the dynamic effects only in body frame, is subject to the integral action discharge drawback during the transient time.

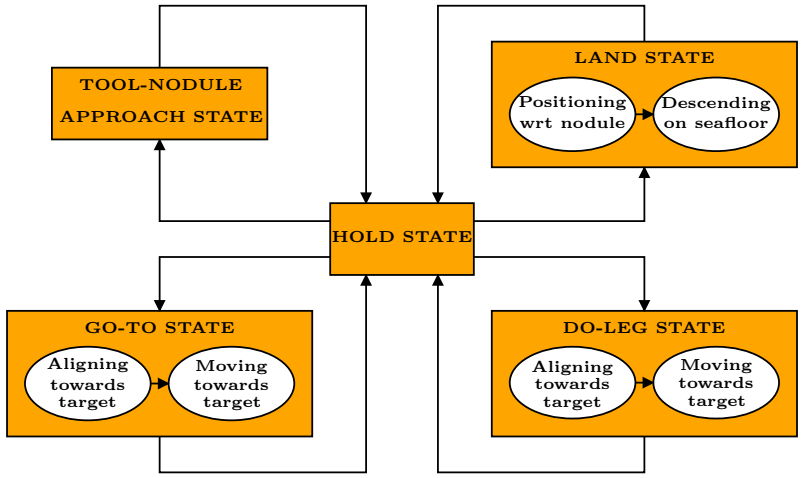

Figure 3.24: The control state machine implemented within the EU funded ROBUST Project: the system exits from the HOLD state, that is the default one, when an external action command is received; then, it reverts to the HOLD state only when the action control objectives are fulfilled.

3.4.2 Considerations

From the preliminary experiments the adaptive control results slightly better than the PI control. However, to evidence the better performance a trajectory in place of a set points should be used. Indeed, as already mentioned before, in presence of large error, e.g., the distance between two way points, the dynamic parameters are essentially integral actions loosing their physical meaning. Then the adaptive control acts as a PI controller. Therefore, aimed at remarking the goodness of the adaptive algorithm, other tests will be done applying a trapezoidal velocity profile. Furthermore, the simulation case studies presented in Section 3.3.1, e.g., movements varying the pitch and yaw angles, will be done to put in evidence the PI drawback in these specific conditions.

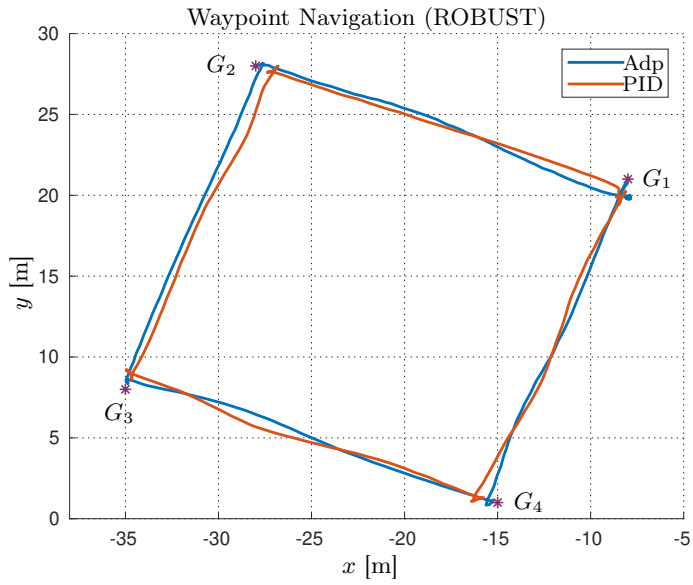

Figure 3.25: ROBUST Project experiments: the UVMS moves from G_1 towards the other desired goal position displaced in a square configuration; both the Adaptive (in blue) and the PI (in red) paths are reported.

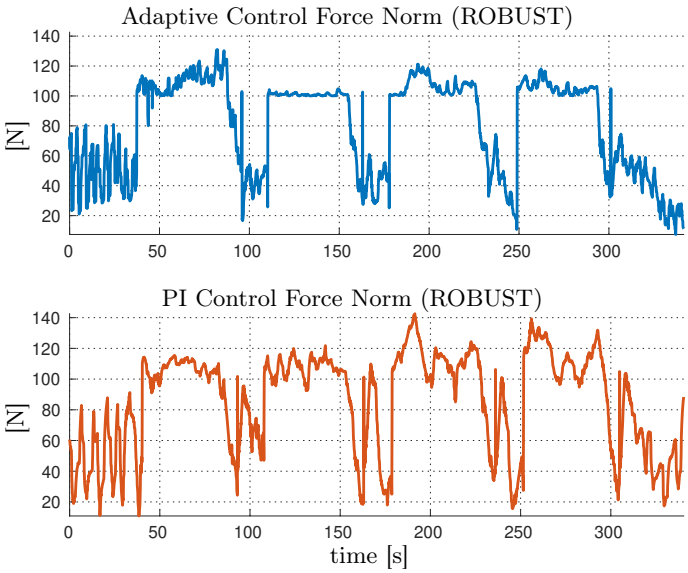

Figure 3.26: ROBUST Project experiments: Adaptive and PI control force norm.

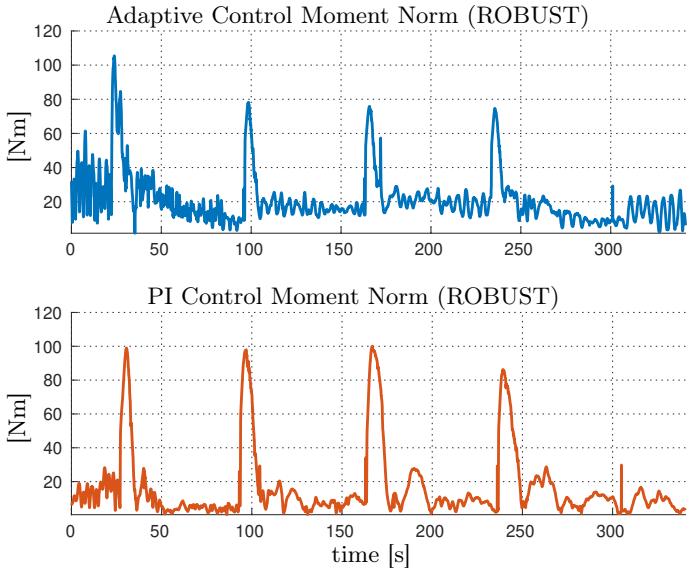

Figure 3.27: ROBUST Project experiments: Adaptive and PI control moment norm.

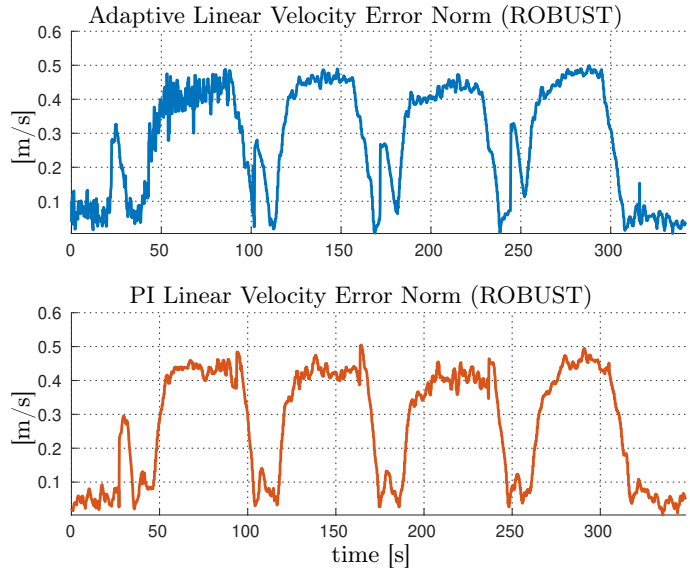

Figure 3.28: ROBUST Project experiments: Adaptive and PI linear velocity error norm.

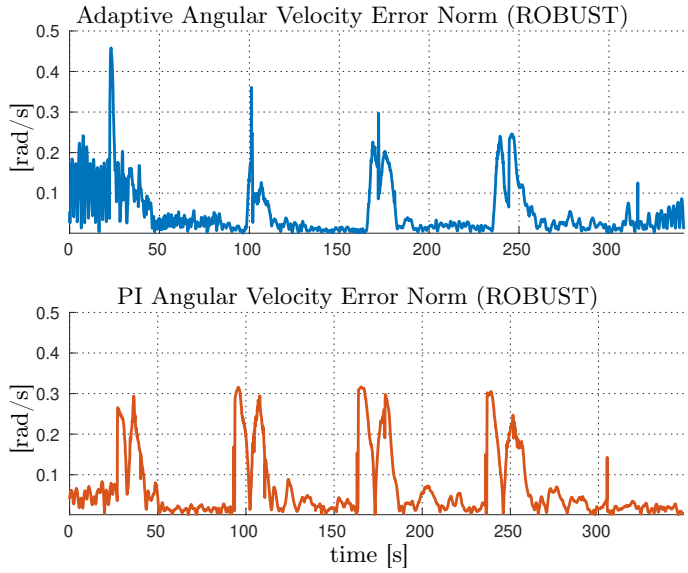

Figure 3.29: ROBUST Project experiments: Adaptive and PI angular velocity error norm.

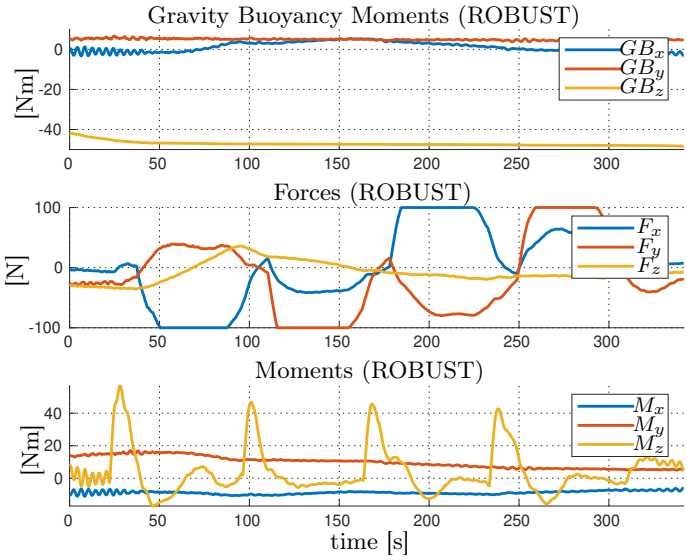

Figure 3.30: ROBUST Project experiments: Adaptive controller dynamic parameters.

Chapter 4

Kinematic Control

4.1 Set-based Task-Priority Inverse Kinematics

The Set-based Task-Priority Inverse Kinematics (STPIK) allows to manage a generic hierarchy $\boldsymbol{\mathcal{H}}$ composed of both set-based and equality-based tasks. In detail, a set-based task can be seen as an equality-based one that is activated/deactivated depending on its current value. The latter can take values in a set which can have a lower and/or upper bound. For each bound several thresholds have to be taken into consideration: activation ($\sigma_{a,l}$ / $\sigma_{a,u}$), safety ($\sigma_{s,l}$ / $\sigma_{s,u}$) and physical (σ_m / σ_M) thresholds, respectively (see Fig. 4.1). In

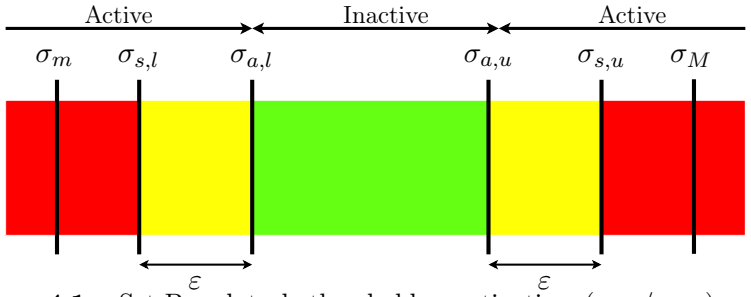

Figure 4.1: Set-Based task thresholds: activation ($\sigma_{a,l}$ / $\sigma_{a,u}$); safety ($\sigma_{s,l}$ / $\sigma_{s,u}$); physical (σ_m / σ_M)

particular, if the set-based task value is in the valid range it is not taken into account. When its value reaches the activation threshold then it is added to the priority hierarchy $\boldsymbol{\mathcal{A}}$, containing only enabled tasks, as an equality-based

task with desired value equal to the safety threshold:

$$\sigma_d = \begin{cases} \sigma_{s,u} & \text{if } \sigma \geq \sigma_{a,u} \\ \sigma_{s,l} & \text{if } \sigma \leq \sigma_{a,l} \end{cases} . \tag{4.1}$$

The set-based task can be deactivated and then removed from the hierarchy only when the solution $\boldsymbol{\zeta}$, computed taking into account all tasks except the set-based one, pushes its value in the valid range. In detail, it is possible to check whether a generic solution $\boldsymbol{\zeta}$ makes a set-based task σ_A go beyond the desired limit or not by evaluating its projection in the task space. Defining $\boldsymbol{J}_A$ as the Jacobian matrix of σ_A, if $\boldsymbol{J}_A\boldsymbol{\zeta} > 0$ the solution would increase the set-based task value, otherwise if $\boldsymbol{J}_A\boldsymbol{\zeta} < 0$ the solution would decrease it. In this way, σ_A can be deactivated if

$$\sigma_A \geq \sigma_{a,u} \ \wedge \ \boldsymbol{J}_A\boldsymbol{\zeta} < 0 \tag{4.2}$$

or

$$\sigma_A \leq \sigma_{a,l} \ \wedge \ \boldsymbol{J}_A\boldsymbol{\zeta} > 0 \, . \tag{4.3}$$

However, the solution is not known a priori and this implies that for each set-based task it is necessary to compute the solution adding and removing the task from the hierarchy. Thus, 2^j solutions have to be computed and stored in a set $\boldsymbol{\mathcal{S}}$ at each algorithm iteration, with j the number of active set-based tasks, as shown in Fig. 4.2.

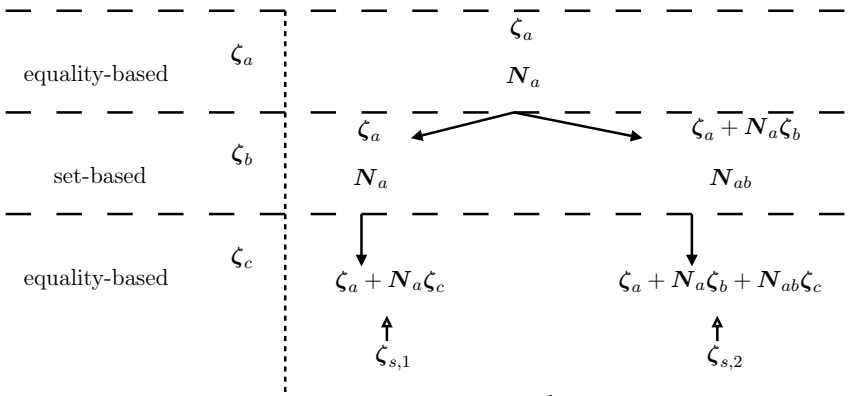

Figure 4.2: Solution tree example: $j = 1 \Rightarrow 2^1$ solutions ($\boldsymbol{\zeta}_{s,1}$, $\boldsymbol{\zeta}_{s,2}$) are generated.

Then, the solutions that satisfy conditions (4.2) or (4.3), among all the solutions in $\boldsymbol{\mathcal{S}}$, have to be selected and stored in a set $\boldsymbol{\mathcal{P}}$. The procedure is described in Algorithm 1.

Algorithm 1: Selection of the solutions that make all the set-based tasks stay in their limits

Data: current Task Hierarchy $\boldsymbol{\mathcal{A}}$ containing n_a set-based tasks, computed solutions $\boldsymbol{\mathcal{S}}$

Result: $\boldsymbol{\mathcal{P}}$ containing all the solutions that push away all the active set-based tasks from their limits, while fulfilling all the equality-based tasks

```
initialize P = ∅, count=0
for all the q̇_j ∈ S do
    for all the set-based tasks σ_k ∈ A do
        if J_k q̇_j > 0   (J_k q̇_j < 0) then
            count++;
            continue;
        else
            break;
        end
    end
    if count==n_a then
        P ← q̇_j
        count=0;
    else
        count=0;
    end
end
```

The final step consists of choosing the highest-norm solution among all the remaining ones in $\boldsymbol{\mathcal{P}}$, as it is the less conservative velocity-wise. The entire algorithm workflow is shown in Fig. 4.3.

As shown in the next Sections, the STPIK algorithm has been used to develop efficient control frameworks.

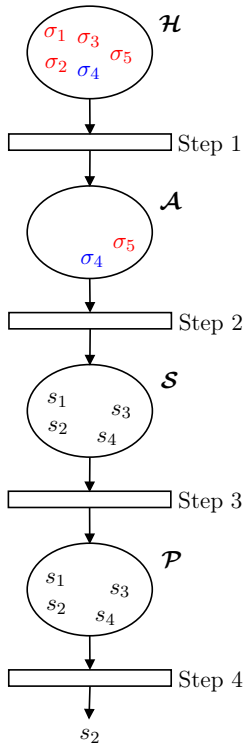

Figure 4.3: Workflow of the algorithm. Red σ_i represents a generic set-based task, while blue σ_i represents a generic equality-based task and s_i is a generic solution. Starting from a generic task hierarchy, the algorithm leads to a unique solution that accomplish simultaneously all the equality-based and the active set-based tasks.

4.2 Analysis and Comparison of Damped Least Square Algorithms

As already mentioned in Section 2.4.4, when the jacobian matrix $\boldsymbol{J}$ is rank deficient the singularities problem arises. To better understand this problem, the Singular Value Decomposition (SVD) of the $\boldsymbol{J}$ is helpful, that is

$$\boldsymbol{J} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}} \;\rightarrow\; \boldsymbol{J}^{\dagger} = \sum_{i=1}^{r} \frac{1}{\sigma_i} \boldsymbol{v}_i \boldsymbol{u}_i^{\mathrm{T}} \tag{4.4}$$

where $\boldsymbol{v}_i \in \mathbb{R}^n$ and $\boldsymbol{u}_i \in \mathbb{R}^m$ are, respectively, the i-th input and output singular vector, and σ_i is the i-th singular value, with $r = \text{rank}(\boldsymbol{J})$ and $\boldsymbol{\sigma}$ ordered so that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$.

It is worth noticing that only in this Section the variable $\boldsymbol{\sigma}$ is the singular value vector whereas in the other Sections is used to represent the generic task variable.

Equation (4.4) allows to recognize that, when a singularity is approached, the r-th singular value tends to zero and a fixed task velocity command along $\boldsymbol{u}_r$ requires joint-space velocities along $\boldsymbol{v}_r$ that grow unboundedly in proportion to the factor $1/\sigma_r$. At the singular configuration, the $\boldsymbol{u}_r$ direction becomes unfeasible for the task variables and $\boldsymbol{v}_r$ adds to the null-space velocities of the arm.

A method to handle this problem is to use the damped least squares, that is a formulation based on weighting the accuracy of tracking the end-effector error with respect to the norm of the joint angle velocity:

$$\|\dot{\boldsymbol{x}}_r - \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}\|^2 + \lambda^2 \|\dot{\boldsymbol{q}}\|^2 \tag{4.5}$$

in which λ is the damping (or weight) factor and

$$\dot{\boldsymbol{x}}_r = \dot{\boldsymbol{x}}_d + \boldsymbol{K}\tilde{\boldsymbol{x}} \in \mathbb{R}^m \tag{4.6}$$

is the reference value vector with $\dot{\boldsymbol{x}}_d$ that is the time derivative of the desired vector for the task function, $\boldsymbol{K}$ is a positive definite (usually diagonal) matrix and $\tilde{\boldsymbol{x}} = \boldsymbol{x}_d - \boldsymbol{x}$ is the task error. The solution to (4.5) is the so called damped pseudoinverse:

$$\boldsymbol{J}^{\dagger^\lambda}(\boldsymbol{q}) = \boldsymbol{J}^{\mathrm{T}}(\boldsymbol{q}) \left(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{q}) + \lambda^2 \boldsymbol{I}\right)^{-1} \tag{4.7}$$

that through the SVD can be rewritten as:

$$\boldsymbol{J}^{\dagger^\lambda} = \sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \lambda^2} \boldsymbol{v}_i \boldsymbol{u}_i^{\mathrm{T}} \,. \tag{4.8}$$

From this last expression it's noticeable that the components for which $\sigma_i \gg \lambda$ are little influenced by the damping factor. On the other hand, when a

singularity is approached, the r-th singular value tends to zero and a fixed task velocity command along $\boldsymbol{u}_r$ requires joint-space velocities along $\boldsymbol{v}_r$ that progressively decrease in proportion to the factor σ_r/λ^2. At the singularity, Eq. (4.4) and (4.8) present a similar behaviour as long as the remaining singular values are significantly larger than the damping factor.

A difficult problem in the use of the damped least squares is the choice of the optimum damping factor λ in all configurations: it should be zero far from singularity and high enough to attenuate joint velocities near singularity. The following algorithms differentiates among them for the configuration in which the damping factor is activated and its value.

Three of these algorithms are based on the singular region estimation by setting a threshold σ_d using the minimum singular value $\sigma_{\min}$ as singularity-closeness measure as discussed in [24]. In these techniques it is noticeable that the value of d is crucial because if it is too high with respect to values of $\sigma_{\min}$, the joint angle velocities are always damped, instead if it is too low there is the risk of high peak velocities close to singularities.

The first approach considered is the one proposed by [46]:

$$\lambda = \begin{cases} 0 & \text{if} \quad \sigma_{\min} \geq \sigma_{d1} \\ \sqrt{\sigma_{\min}(\sigma_{d1} - \sigma_{\min})} & \text{if} \quad \sigma_{\min} < \sigma_{d1} \end{cases} \tag{4.9}$$

where

$$\sigma_{d1} = \frac{1}{\|\dot{\boldsymbol{q}}\|_{\max}} \tag{4.10}$$

with

$$\|\dot{\boldsymbol{q}}\|_{\max} \geq \left\| \boldsymbol{J}^{\dagger^{\lambda}} \dot{\boldsymbol{x}}_r \right\| . \tag{4.11}$$

The threshold σ_{d1} is set in function of $\|\dot{\boldsymbol{q}}\|_{\max}$ that is the boundary on the norm of the solution $\dot{\boldsymbol{q}}$ and depends on the specific manipulator joint velocity limits. Equation (4.9) is represented in Fig. 4.4 where it's noticeable how λ becomes null when $\sigma_{\min}$ approaches zero.

The second method analysed is by Arati S.Deo and Ian D.Walker [28] that is not based on any measure of close-to-singularity region. It is an iterative

technique that computes the optimal damping factor when $\|\dot{\boldsymbol{q}}\| > \|\dot{\boldsymbol{q}}\|_{\max}$ with the following iteration:

$$\lambda_{k+1} = \lambda_k - \left[\frac{\left\|\dot{\boldsymbol{q}}_{\lambda_k}\right\|}{\|\dot{\boldsymbol{q}}\|_{\max}}\right]\left[\frac{\phi(\lambda_k)}{\phi'(\lambda_k)}\right], \tag{4.12}$$

where

$$\phi(\lambda_k) = \left\|\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{J}\boldsymbol{J}^{\mathrm{T}} + \lambda_k \boldsymbol{I})^{-1}\dot{\boldsymbol{x}}_r\right\| - \|\dot{\boldsymbol{q}}\|_{\max} \tag{4.13}$$

that using the SVD can be rewritten as:

$$\phi(\lambda_k) = \left\|\boldsymbol{\Sigma}^{\mathrm{T}}(\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{\mathrm{T}} + \lambda_k \boldsymbol{I})^{-1}\boldsymbol{U}^{\mathrm{T}}\dot{\boldsymbol{x}}_r\right\| - \|\dot{\boldsymbol{q}}\|_{\max} \tag{4.14}$$

and letting $\boldsymbol{U}^{\mathrm{T}}\dot{\boldsymbol{x}}_r = [\gamma_1\, \gamma_2 \ldots \gamma_r]^{\mathrm{T}}$ it becomes

$$\phi(\lambda_k) = \sqrt{\sum_i^r \frac{\sigma_i^2\gamma_i^2}{(\sigma_i^2 + \lambda_k)^2}} - \|\dot{\boldsymbol{q}}\|_{\max}\ . \tag{4.15}$$

$\phi'(\lambda_k)$ is obtained differentiating $\phi(\lambda_k)$ with respect to λ_k:

$$\phi'(\lambda_k) = -\frac{1}{\left\|\dot{\boldsymbol{q}}_{\lambda_k}\right\|}\sum_i^r \frac{\sigma_i^2\gamma_i^2}{(\sigma_i^2 + \lambda_k)^2}\ . \tag{4.16}$$

Thus with this method the optimal λ^* such that $\phi(\lambda^*) = 0$
$\Rightarrow \|\dot{\boldsymbol{q}}_{\lambda^*}\| = \|\dot{\boldsymbol{q}}\|_{\max}$ is obtained.

The third method analysed is by F. Caccavale, S. Chiaverini and B. Siciliano [16]:

$$\lambda^2 = \begin{cases} 0 & \text{if} \quad \sigma_{\min} \geq \sigma_{d2} \\ \left[1 - \left(\frac{\sigma_{\min}}{\sigma_{d2}}\right)^2\right]\lambda_M^2 & \text{if} \quad \sigma_{\min} < \sigma_{d2} \end{cases} \tag{4.17}$$

where σ_{d2}, as σ_{d1} in the Maciejewski algorithm, defines the size of the singular region and λ_M is the final value for the damping factor that is when $\sigma_{\min} = 0$ then $\lambda = \lambda_M$. In this case the choice of σ_{d2} and λ_M is not explicit: they are heuristically chosen according to the specific manipulator used. From a geometric point of view it is possible to observe that (4.17) represents a quarter of ellipse as shown in Fig. 4.4.

The fourth method analysed is the one by [11]:

$$\lambda = \begin{cases} 0 & \text{if} \quad \sigma_{\min} \geq \sigma_{d3} \\ \sqrt{\sigma_{\min}(\sigma_{d3} - \sigma_{\min})} & \text{if} \quad \sigma_{d3}/2 \leq \sigma_{\min} < \sigma_{d3} \\ \sigma_{d3}/2 & \text{if} \quad \sigma_{\min} < \sigma_{d3}/2 \end{cases} \tag{4.18}$$

where

$$\sigma_{d3} = \frac{\|\dot{\boldsymbol{x}}_r\|}{\|\dot{\boldsymbol{q}}\|_{\max}} \tag{4.19}$$

with $\|\dot{\boldsymbol{q}}\|_{\max}$ already defined in Eq. (4.11). The threshold σ_{d3} is now dependent on both $\dot{\boldsymbol{x}}_r$, that includes the error $\tilde{\boldsymbol{x}}$, and $\|\dot{\boldsymbol{q}}\|_{\max}$ and so it is set dynamically. Examining Eq. (4.18) shown in Fig. 4.4 it's clear that, unlike Maciejewski, close to a singular configuration the damping factor λ is set to $\sigma_{d3}/2$.

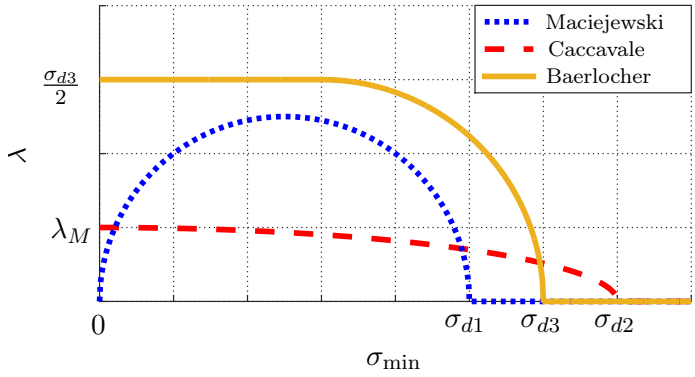

Figure 4.4: Choice of the damping factor λ according to Maciejewski's, Baerlocher's and Caccavale's algorithms.

A variant of Baerlocher's algorithm is based on finding iteratively the optimal λ for a given boundary $b_{\max}$ with Newton's method. In details, for each sample, when inside the singular region and then with λ non null, the following iterative law is used:

$$\lambda_{k+1} = \lambda_k + \alpha(\|\dot{\boldsymbol{q}}\|_{\max} - \|\dot{\boldsymbol{q}}_k\|) \tag{4.20}$$

where k represents the k-th iteration and α is a gain factor. The iterations number and the gain α are 2 free parameters that need to be suitably set.

The last method analysed is by [73] that is different from the previous ones because it does not define a singular region where to apply the damping factor but it proposes to use always small biasing values even if not close to a singular configuration. In particular the following Jacobian pseudoinverse is applied:

$$\boldsymbol{J}^{\dagger} = \left(\boldsymbol{J}^{\mathrm{T}}\mathbf{W}_E\boldsymbol{J} + \mathbf{W}_N\right)^{-1}\boldsymbol{J}^{\mathrm{T}}\mathbf{W}_E\dot{\boldsymbol{x}}_r \tag{4.21}$$

where $\boldsymbol{W}_E \in \mathbb{R}^{m\times m}$ is the weighted matrix, that reflects the workspace components priorities, and the matrix $\boldsymbol{W}_N \in \mathbb{R}^{n\times n}$ represents the damping factor defined as:

$$\boldsymbol{W}_N = E\boldsymbol{I} + \overline{\boldsymbol{W}}_N \tag{4.22}$$

with $E = \frac{1}{2}\dot{\boldsymbol{x}}_r^{\mathrm{T}}\boldsymbol{W}_E\dot{\boldsymbol{x}}_r$, $\boldsymbol{I} \in \mathbb{R}^{n\times n}$ is the identity matrix and $\overline{\boldsymbol{W}}_N \in \mathbb{R}^{n\times n}$ is a diagonal matrix of small biasing values $\overline{w}_N$. According to an empirical knowledge it is proposed to set $\overline{w}_N \simeq 1.0\times 10^{-2}l^2 \sim 1.0\times 10^{-3}l^2$ where l is the length of a typical link, $l = 0.1 \sim 100\,\mathrm{m}$.

4.2.1 DLS Algorithms Simulation

All the algorithms previously shown have been evaluated for comparison. In details, considering physically achievable tasks and not taking into consideration high tracking precision during the transient, each algorithm results satisfactory. Anyway in this work the following metrics have been considered:

I chattering absence;

II high tracking precision;

III exact joint velocity saturation;

IV trajectory independence,

and the two following study cases have been determined:

Case 1: a boundary singularity obtained with a desired end-effector position outside the workspace, that represents a desired task not physically achievable;

Case 2: an internal singularity obtained with a desired position inside the workspace, that represents a physically achievable desired task.

The evaluation tests have been implemented using the 7 DOFs Kinova Jaco^2 manipulator, the first version without the spherical wrist in the specific case (Fig. 4.5).

Figure 4.5: Kinova Jaco^2 manipulator.

The end-effector position control has been implemented for the comparison tests through the CLIK scheme with gain $\boldsymbol{K} = \text{diag}\{1, 1, 1\}$, assigning a trapezoidal velocity profile with a trajectory time $T_f = 10\,\text{s}$ and using a sampling rate $T_s = 0.01\,\text{s}$. The boundary $\|\dot{\boldsymbol{q}}\|_{\max}$ has been set equal to the minimum joint velocity limit to ensure that no other greater limit was passed, i.e., $\|\dot{\boldsymbol{q}}\|_{\max} = 0.6283\,\text{rad/s}$.

The Caccavale's algorithm, which gives good results if properly calibrated, has not been taken into consideration in simulations because it is based on parameters that strongly depend on the specific robot. Instead the present work wants to focus on the algorithms that claim to be independent from the robotic system.

Maciejewski Algorithm

In the first case (outside workspace) the joint velocities shown in Fig. 4.6(a) were obtained. The resulting chattering is due to Eq. (4.9) because when $\sigma_{\min}$

approaches zero also λ goes to zero (as shown in Fig. 4.6(d)) thus the damping factor's function fails. To better understand, the SVD of Eq. (4.11) can be useful:

$$\left\| \boldsymbol{J}^{\dagger^{\lambda}} \dot{\boldsymbol{x}}_r \right\| = \left\| \left(\sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \lambda^2} \boldsymbol{v}_i \boldsymbol{u}_i^{\mathrm{T}} \right) \dot{\boldsymbol{x}}_r \right\| \leq b_{\max} \tag{4.23}$$

From this formulation it's clear that if the error approaches zero the relative joint velocity components go to zero too. Anyway if the error is not null, as in our tests, a damping factor λ, that is much greater than the minimum singular value $\sigma_{\min}$, is necessary to bring to zero the fraction $\frac{\sigma_i}{\sigma_i^2+\lambda^2}$ and so the entire expression in Eq. (4.23) and accordingly the relative joint velocity components. Fig. 4.6(e) shows the end-effector position error norm.

In the second case (inside the workspace), since the error approaches zero, joint velocities do not present chattering, as shown in Fig. 4.6(b). Anyway the threshold σ_{d1} is fixed and therefore the damping is present even if it is not strictly necessary (observable in Fig. 4.6(d)). This causes a noticeable reduction of the tracking precision as shown in Fig. 4.6(e).

Deo and Walker Algorithm

Simulation results obtained by applying Deo and Walker's algorithm are shown in Fig. 4.7. In particular it is noticeable that this method is not able to manage the first case (outside the workspace). In fact joint velocities present chattering as shown in Fig. 4.7(a). This is due to the computation of the optimal damping factor which depends on $\phi(\lambda)$ defined in Eq. (4.13). This means that joint velocities are damped only if their norm passes $\|\dot{\boldsymbol{q}}\|_{\max}$ without considering other parameters, first of all the error.

In the second case (inside workspace), where the error approaches zero, the algorithm is able to manage the singularity and to fill $\|\dot{\boldsymbol{q}}\|_{\max}$ giving a good error tracking precision as shown in Fig. 4.7(c).

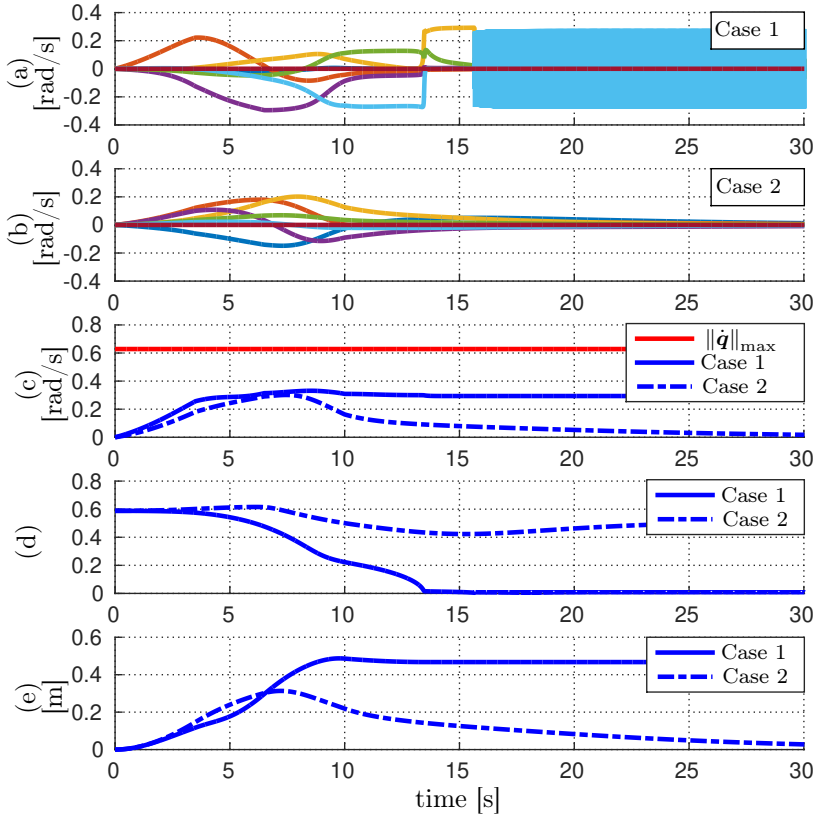

Figure 4.6: Maciejewski's algorithm: (a),(b) joint velocities; (c) joint velocities norm; (d) damping factor λ; (e) end-effector position error norm.

Baerlocher Algorithm

Fig. 4.8 shows the results obtained by Baerlocher's algorithm. Unlike Maciejewski, it avoids the chattering in the joint velocities, as shown in Fig. 4.8(a), thanks to the different damping factor in Eq. (4.18). Furthermore, using a dynamic threshold σ_{d3} linked to the task-error the joint velocity norm is able to fill more the $\|\dot{\boldsymbol{q}}\|_{\max}$ with respect to Maciejewski. This means a better tracking error during the trajectory. Anyway it is not sufficient since, as shown in Fig. 4.8(c) and 4.8(d), the joint velocities norm never fills completely the $\|\dot{\boldsymbol{q}}\|_{\max}$. This issue underlines how the joint velocities norm does not generally represent the

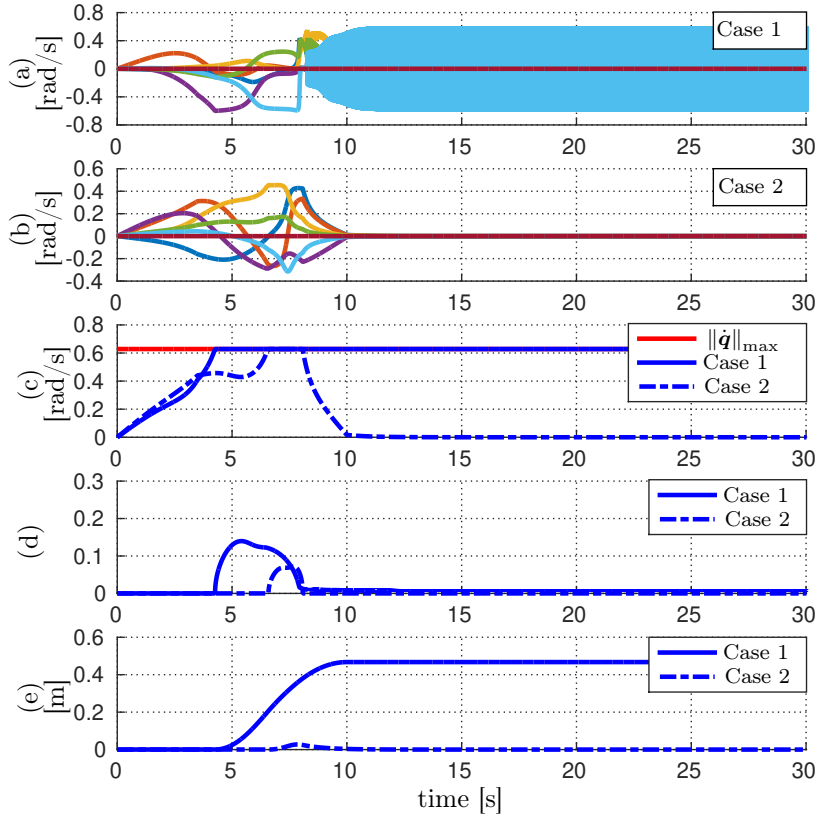

Figure 4.7: Deo and Waker's algorithm: (a),(b) joint velocities; (c) joint velocities norm; (d) damping factor λ; (e) end-effector position error norm.

real constraint to take into consideration for a robotic system. Thus there is still the necessity to obtain a better error-tracking.

Another test has been executed putting in evidence a particular behaviour. More in details this algorithm has been tested again considering the second case (inside the workspace) setting the gain $\boldsymbol{K} = \text{diag}\{50, 50, 50\}$. Fig. 4.9 shows that joint velocities have a high growth in proximity of the close-to-singularity configuration but even after. This is due to the tracking error (observable in Fig. 4.9(d)) that influences the threshold σ_{d3} (defined in Eq. (4.19)). In fact

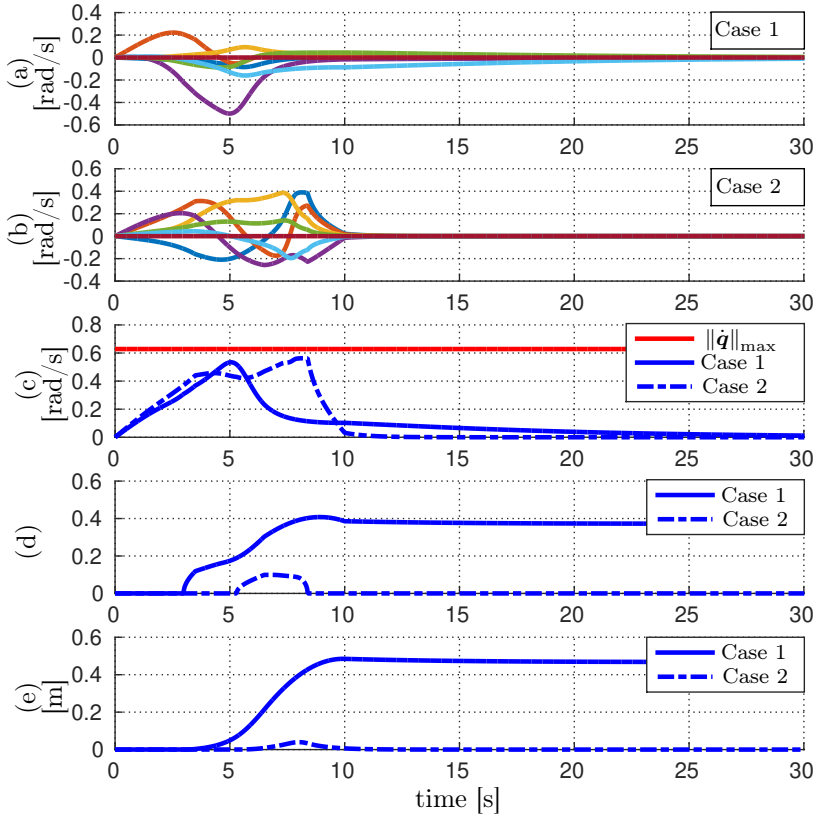

Figure 4.8: Baerlocher's algorithm: (a), (b) joint velocities; (c) joint velocities norm; (d) damping factor λ; (e) end-effector position error norm.

during the transition through the close-to-singularity region the error is accumulated because of the damping (shown in Fig. 4.9(c) and 4.9(d)). Therefore the CLIK dynamics tries to set to zero the error causing high joint velocities. This puts in evidence how the CLIK gain influences the computation of the damping factor in the Baerlocher's algorithm.

Iterative Baerlocher Algorithm

Simulation results of the Baerlocher's variant are shown in Fig. 4.10. This variant requires to set the gain α and the iterations number. For both the cases

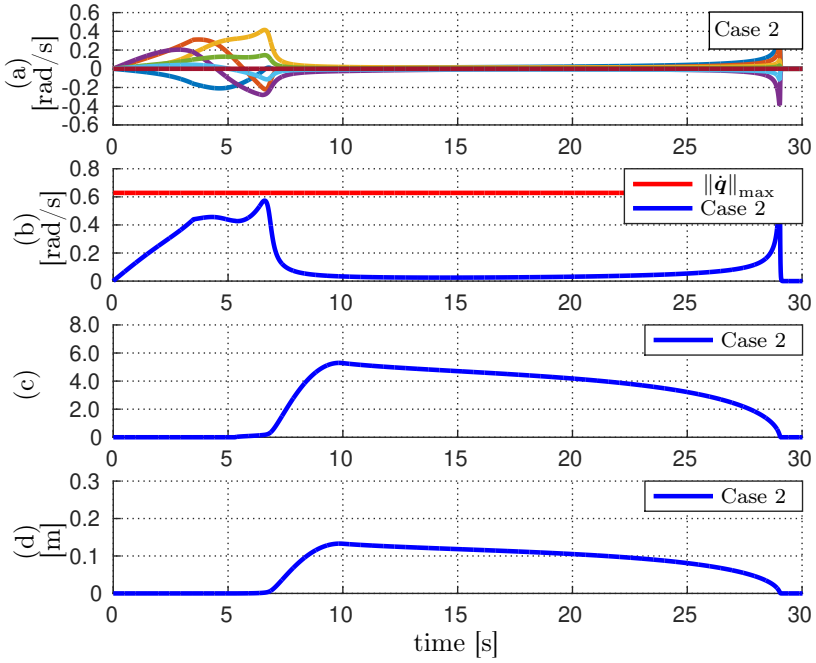

Figure 4.9: Baerlocher's algorithm with $\boldsymbol{K} = \text{diag}\{50, 50, 50\}$: (a) joint velocities; (b) joint velocities norm; (c) damping factor λ; (d) end-effector position error norm.

(outside/inside) were set 40 iterations and $\alpha = 10^{-2}$. As Fig 4.10(c) shows, the joint velocities norm is able to fill the $\|\dot{\boldsymbol{q}}\|_{\max}$ ensuring a better tracking error. Anyway this algorithm requires an appropriate parameters setting to obtain good results and it is also important to consider the computational payload, because each sampling step a certain number of iterations is executed.

Other tests have been made on this variant of Baerlocher's method showing its task dependency. In details, starting by setting $\|\dot{\boldsymbol{q}}\|_{\max} = 1$, the gain $\alpha = 10^{-2}$ and 40 iterations, Fig. 4.11(a) represents the joint velocity norm obtained with desired position $x_{d1} = [2.5, 0, 0.4]$ and the ones obtained with desired position $x_{d2} = [0, 0, 1.5]$. Thus, this makes noticeable how the same parameters used

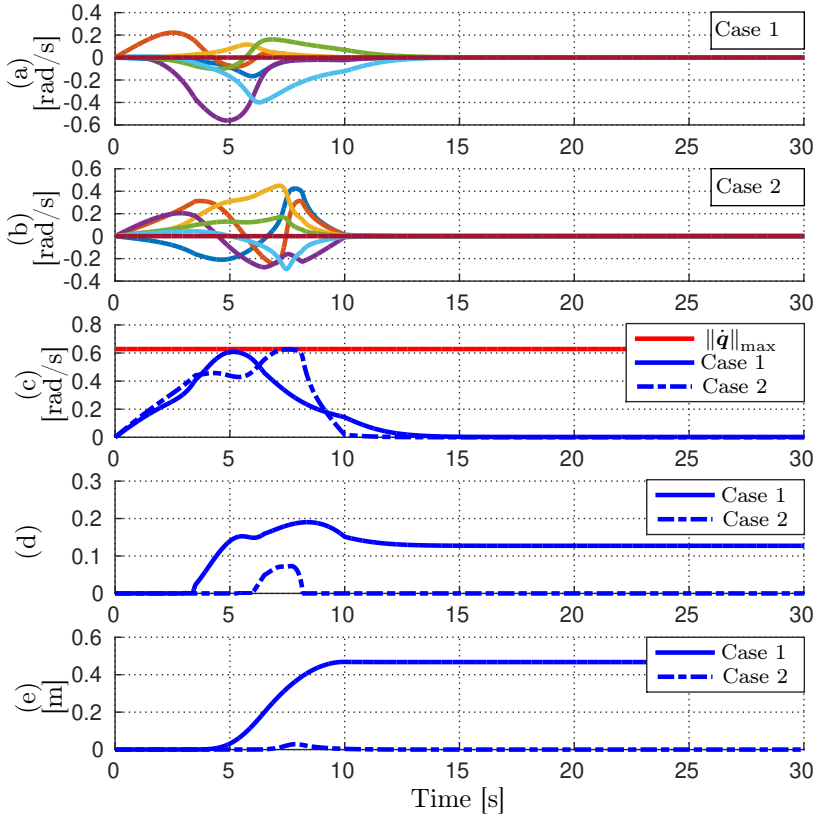

Figure 4.10: Iterivative Baerlocher's algorithm: (a),(b) joint velocities; (c) joint velocities norm; (d) damping factor λ; (e) end-effector position error norm.

in the first case then are not suitable in the second configuration. This means that the algorithm is configuration-dependent and it can not be used with the same parameters in general but it needs to be calibrated each time.

Sugihara Baerlocher Algorithm

Fig. 4.12 shows the simulation results obtained by Sugihara's algorithm. More in detail, it was tested setting the bias values $\overline{w}_N = 10^{-2}$. As it's possible to observe from Fig. 4.12(c), the constant presence of the matrix $\overline{\boldsymbol{W}}_N$ in Eq. (4.22)

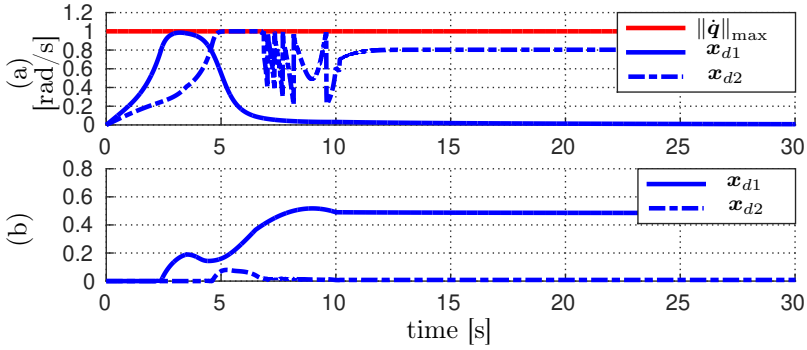

Figure 4.11: Iterative Baerlocher's algorithm with $\|\dot{\boldsymbol{q}}\|_{\max} = 1$: (a) joint velocities norm; (b) damping factor λ.

(that in this method represents the damping factor) even if the manipulator is not in a close-to-singularity configuration, causes a higher error in tracking trajectory with the impossibility to track the desired end-effector velocity satisfactorily. Moreover, this method does not allow to set a $\|\dot{\boldsymbol{q}}\|$ either and, as stated previously, the setting of $\overline{w}_N$ is left to an empirical knowledge.

DLS Algorithms Considerations

Taking into account the above simulation results, the following considerations can be drawn:

- Maciejewski: because of Eq. (4.9) it is not able to manage cases where the error $\tilde{\boldsymbol{x}}$ is not null (for example outside workspace desired position) causing chattering in the joint velocities. Furthermore the threshold d of the close-singular region is fixed and therefore it results very conservative.
- Caccavale: it does not present a general way for setting the threshold d and the maximum value of lambda λ_M but it requires heuristics.
- Baerlocher: it allows to set a maximum boundary ($b_{\max}$) for the joint velocities norm that is always respected. Anyway it presents a crucial factor that drastically influences its performance: it is the threshold d. In

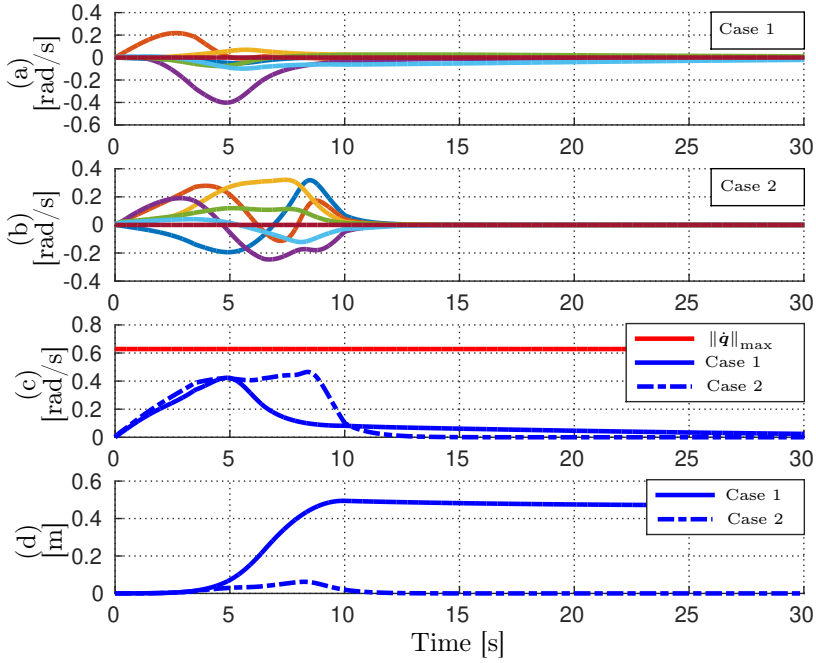

Figure 4.12: Sugihara's algorithm: (a),(b) joint velocities; (c) joint velocities norm; (d) end-effector position error norm.

fact if d is too high, regardless of the cause, the damping factor is activated too early even if it is not necessary, causing an increasing tracking error. As consequence it is possible to have not null error and an user that thinks to properly manage a singularity setting a high threshold, can have bad results. Furthermore there is a link between the CLIK gain and the algorithm performances that has to be taken into consideration.

- Iterative Baerlocher: its performance is linked to the iterations number and the gain α. Good parameters for a specific configuration could not be suitable for another one (as shown in tests). Furthermore the iterations number increment the computational load and this can create delay on the control loop frequency depending on the hardware.

- Sugihara: its limit is represented by the constant presence of the bias

values causing a continuous damping. This issue can be solved increasing the gain $\boldsymbol{K}$. Anyway the impossibility to set a maximum boundary ($b_{\max}$) makes very easy to violate the joint velocities limits (that present saturations and therefore don't respect the desired velocities increasing the tracking error).

- Deo and Walker: it is able to fill the $b_{\max}$ independently on any measure of singularity-closeness and therefore to manage the singularity only if the error approaches zero otherwise it provides velocities characterized by chattering.

Furthermore all these algorithms present a common negative constraint: they are based on the joint velocities norm. Instead, especially for anthropomorphous manipulators, the real constraint is on velocity of the single joints because they present different velocity limits, in general lower velocities to the shoulder joints and higher velocities to the wrist ones are allowed.

Let us recall here the metrics previously defined: **I** chattering absence; **II** high tracking precision; **III** exact joint velocity saturation; **IV** trajectory independence. Then, considerations on all algorithms can be drawn:

	Case	I	II	III	IV
Maciejewski	1	No	No	No	-
Maciejewski	2	Yes	No	No	-
Deo and Walker	1	No	Yes	Yes	-
Deo and Walker	2	Yes	Yes	Yes	-
Baerlocher	1	Yes	No	No	-
Baerlocher	2	Yes	No	No	-
Iter. Baerlocher	1	Yes	Yes	Yes	No
Iter. Baerlocher	2	Yes	Yes	Yes	No
Sugihara	1	Yes	No	No	-
Sugihara	2	Yes	No	No	-

Table 4.1: Table summarizing the algorithms' performances

The trajectory independence has been verified only for the Iterative Baerlocher's algorithm because it has been the only one to satisfy the metrics I, II and III.

4.3 Assistive Control Framework for ROVs

ROV guidance and coordination requires a very high-experienced human operator, and a concentration level that grows depending on the environmental conditions, e.g., visibility and ocean currents [26]. On average, in case of bad work conditions, an operator is able to drive the system for 30 minutes at most. Contrarily, in case of good operative conditions, he can work without pauses for one hour and half. Thus, the presence of automatism, e.g., station-keeping, auto-heading, way-point navigation and auto-depth, that helps the operator in performing some tasks, can improve the mission performances. Furthermore control tasks as obstacle avoidance and auto-altitude can ensure to avoid vehicle collisions even when, in case of bad operative conditions, the human operator could fail.

On the basis of real needs that arose from a survey realized with different professional ROV pilots, an assistive control framework has been developed. In particular, the survey results remarked the following situations:

1) *Unbalanced vehicle.* In several ROV designs, the vehicle is under actuated leaving roll and pitch to be stabilized by a proper weight distribution. However, moving masses such as manipulators or a wrong payload positioning may cause the vehicle to exhibit non-null roll and/or pitch at steady state. For example, this is what has happened within the framework of the H2020 European Project DexROV [29] where an existing under actuated vehicle has been customized, and the presence of two manipulators and of the perception system has caused the system to operate with a non-null pitch angle depending on the arm configuration. In such a case, in order to perform a movement along the earth-fixed frame surge direction (the green dashed line in Fig. 4.13), the operator, having control of the body-fixed variables, naturally follows a saw tooth shaped path (the red dashed line in Fig. 4.13) by alternating velocity commands to the horizontal and vertical thrusters respectively. In this sense, a control task allowing the operator to send velocity commands along the effective desired direction might improve the mission efficiency. Even when the vehicle is fully actuated, it might be appropriate to leave the roll and pitch angles not actuated to save energy during long missions. In such a case, the situation is analogous

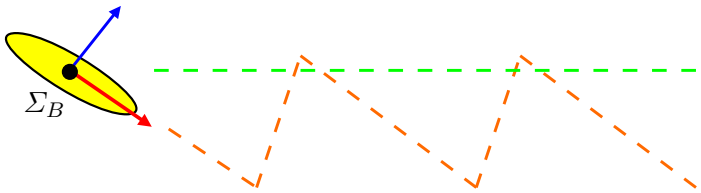

Figure 4.13: Vehicle with non-null pitch angle: the green dashed line represents the desired path; the red dashed one is the path followed by an operator acting on the body-fixed variables.

to the above and this feature might help the operator. Moreover, the possibility to enable/disable the control task according to the operator needs during the mission is required.

2) *Altitude/depth control.* Controlling depth and altitude is intrinsically a conflicting requirement. The operator is often required to travel at constant depth (orange dashed line in Fig. 4.14), however, for safety reasons a minimum altitude needs to be maintained. Auto-depth is an existing control feature in most of the vehicles, however, the operator needs to keep track of the current altitude to avoid of impacting the sea bottom. A required feature is an automatic switching between auto-depth and auto-altitude without human intervention.

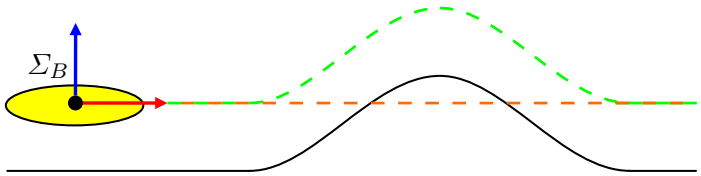

Figure 4.14: The green dashed line is the path followed through an auto-altitude task; the orange dashed one is the path followed through an auto-depth task. Operators might be helped by an automatic switching between the two tasks.

The need for this feature is emphasized also by considering again the unbalanced vehicle case.

3) *Reference Frames.* Depending on the specific operation to perform, it could be very helpful for the operator to have the possibility to choose to command the ROV in inertial frame or in the body-fixed frame, e.g., for intervention operation as manoeuvring valves of a panel (see Fig. 4.15).

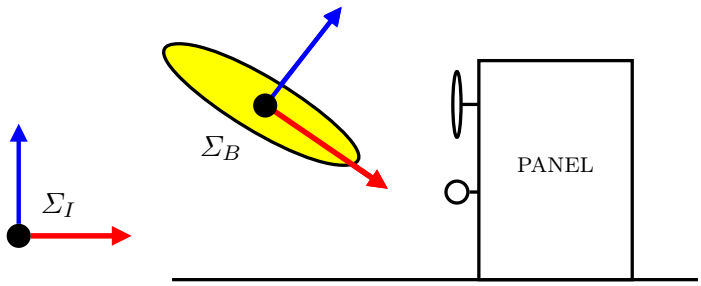

Figure 4.15: Reference frames: possibility for the operator to provide commands in inertial or body-fixed frame depending on the manoeuvring in front of the panel.

4.3.1 Implemented tasks

With the aim to support the operator in guiding and manoeuvring underwater vehicles, different tasks $\boldsymbol{\sigma}_i$ have been implemented with the corresponding Jacobians $\boldsymbol{J}_i \in \mathbb{R}^{m\times 6}$. In detail, tasks are divided into equality-based and set-based, respectively. Starting from the equality-based ones there are:

• **vehicle position** ($m = 3$). It consists in the control of vehicle position. Thus the task is defined as $\boldsymbol{\sigma}_i = \boldsymbol{\eta}_1 \in \mathbb{R}^3$ and the relative Jacobian is $\boldsymbol{J}_i = \boldsymbol{J}_{pos} \in \mathbb{R}^{3\times 6}$ where

$$\boldsymbol{J}_{pos} = \begin{bmatrix} \boldsymbol{R}_0^B & \boldsymbol{0}_{3\times 3} \end{bmatrix} . \tag{4.24}$$

• **vehicle orientation** ($m = 3$). The vehicle orientation control is expressed by $\boldsymbol{\sigma}_i = \boldsymbol{\eta}_2 \in \mathbb{R}^3$ with the Jacobian defined as $\boldsymbol{J}_i = \boldsymbol{J}_{ori} \in \mathbb{R}^{3\times 6}$ where

$$\boldsymbol{J}_{ori} = \begin{bmatrix} \boldsymbol{0}_{3\times 3} & \boldsymbol{J}_o \end{bmatrix} . \tag{4.25}$$

• **vehicle pose** ($m = 6$). This task allows to control the entire vehicle configuration merging the previous two tasks. Therefore it is defined as $\boldsymbol{\sigma}_i = \boldsymbol{\eta} \in \mathbb{R}^6$ with

$$\boldsymbol{J}_i = \begin{bmatrix} \boldsymbol{J}_{pos}^{\mathrm{T}} & \boldsymbol{J}_{ori}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{6\times 6} . \tag{4.26}$$

• **vehicle attitude** ($m = 2$). It consists in the control of two components of orientation, in detail the roll and pitch ones. Thus $\boldsymbol{\sigma}_i = [\phi \quad \theta]^{\mathrm{T}} \in \mathbb{R}^2$ with the Jacobian $\boldsymbol{J}_i$ defined as the first two rows of the $\boldsymbol{J}_{ori}$ matrix. It results very useful if the operator needs to work with specific values of roll and pitch angles or in case the ROV is not well balanced.

• **vehicle relative field of view** ($m = 1$). This task allows to direct the vehicle towards a desired target $\boldsymbol{p}_O$, e.g., a panel on the seabed. Thus if the human operator knows the target position a priori, he can activate this task without manually manoeuvring the vehicle. From the control perspective, it is not necessary to control all orientation DOFs but only the desired outgoing vector [53]. In detail, the task function is defined as $\sigma_i = \sigma_{FoV}$ with

$$\sigma_{FoV} = \pi/2 - \arccos\big(r_k^E/||\boldsymbol{r}^E||\big)\,, \tag{4.27}$$

where $\boldsymbol{r}^E = \boldsymbol{R}_0^B\big(\boldsymbol{\eta}_1 - \boldsymbol{p}_O\big) \in \mathbb{R}^3$ and r_k^E is the k-th component of $\boldsymbol{r}^E$ corresponding to the desired outgoing vector. The relative Jacobian is $\boldsymbol{J}_i = \boldsymbol{J}_{FoV} \in \mathbb{R}^{1\times 6}$ with

$$\boldsymbol{J}_{FoV} = \frac{\boldsymbol{e}_k^{\mathrm{T}}}{\sqrt{||\boldsymbol{r}^E||^2 - |r_k^E|^2}}\big[\boldsymbol{P}_r^{\perp}\boldsymbol{R}_0^B \quad \boldsymbol{S}(\boldsymbol{r}^E)\big]\boldsymbol{J}\,, \tag{4.28}$$

where $\boldsymbol{P}_r^{\perp} = \boldsymbol{I}_3 - \boldsymbol{r}^E\big(\boldsymbol{r}^E\big)^{\mathrm{T}}/||\boldsymbol{r}^E||^2 \in \mathbb{R}^{3\times 3}$, $\boldsymbol{S}(\boldsymbol{r}^E)$ is the skew-symmetric matrix and e_k is the k-th unit vector of the canonical base spanning $\mathbb{R}^3$.

• **vehicle heading** ($m = 2$). This task, contrarily to the vehicle attitude, controls the pitch and yaw components allowing to maintain the ROV along a desired straight trajectory. It is defined as $\boldsymbol{\sigma}_i = [\theta \quad \psi]^{\mathrm{T}} \in \mathbb{R}^2$ and the corresponding Jacobian $\boldsymbol{J}_i$ is defined as the last two rows of the $\boldsymbol{J}_{ori}$ matrix.

• **vehicle depth** ($m = 1$). Most of underwater vehicles is equipped with depth sensors. Indeed they are pressure-based and therefore they are more reliable than altitude ones that are instead based on acoustic systems. Thus a control on ROV depth is implemented giving the possibility to the operator to set a determined depth value. In detail, this task is defined as $\sigma_i = z$ and its Jacobian $\boldsymbol{J}_i$ corresponds to the third row of the $\boldsymbol{J}_{pos}$ matrix.

The set-based tasks, on the other side, are the following:

• **vehicle roll, pitch and yaw limit**, respectively ($m = 1$). ROVs generally have small variance ranges for roll and pitch components. This is essentially due to hardware that presents physical limits for the orientation. Contrarily, the yaw component can present a large variance range or also it can have no limits. Indeed, the only constraint is represented by the presence of the tether. The limits can be set directly by the operator. The tasks and relative Jacobians are defined as the first, the second and the third row of the $\boldsymbol{\eta}_2$ vector and $\boldsymbol{J}_{ori}$ matrix respectively.

• **vehicle altitude limit** ($m = 1$). The following task allows to set a minimum altitude value from seabed and eventual objects preventing possible damages to the vehicle. It is defined as $\sigma_i = z$ with the corresponding Jacobian $\boldsymbol{J}_i$ equal to the third row of $\boldsymbol{J}_{pos}$ matrix.

• **obstacle avoidance** ($m = 1$). The obstacle avoidance ensures of avoiding eventual obstacles along the trajectory without the operator intervention who has just to set the safety distance. This task can be implemented controlling the ROV distance from the obstacle position $\boldsymbol{p}_{ob} \in \mathbb{R}^3$. In detail, it is defined as

$$\sigma_i = \sqrt{(\boldsymbol{p}_{ob} - \boldsymbol{\eta}_1)^{\mathrm{T}}(\boldsymbol{p}_{ob} - \boldsymbol{\eta}_1)} \in \mathbb{R}^1 \tag{4.29}$$

with the corresponding Jacobian

$$\boldsymbol{J}_i = -\frac{(\boldsymbol{p}_{ob} - \boldsymbol{\eta}_1)^{\mathrm{T}}}{||\boldsymbol{p}_{ob} - \boldsymbol{\eta}_1||}\boldsymbol{J}_{pos} \in \mathbb{R}^{1\times 6}\ . \tag{4.30}$$

All described tasks are dynamically combined through the priority hierarchy resorting to the set-based task-priority inverse kinematics control.

4.3.2 Control Framework Architecture

The assistive control framework, shown in Fig. 4.16, has been developed inside the well-known Robot Operating System (ROS). Thus, the implemented software is structured in a graph architecture where processes are able to communicate among them like nodes of the same network. In particular, the framework

architecture is divided into two main blocks: the command station and the vehicle simulator.

The Command Station is represented by the graphical user interface that allows the operator to send commands by joystick and to enable/disable the different tasks. It is worth noticing that the task enabling/disabling is different from the set-based task activation/deactivation. Indeed, when a task is enabled/disabled by the operator it is added/removed to/from the control task hierarchy. On the other side, the task activation/deactivation, as described in section 4.1, is a crucial step of the set-based control algorithm, and therefore it is not managed by the operator.

The Vehicle Simulator block, developed in Gazebo environment, consists of the implemented high and low level controllers, and the vehicle kinematic/dynamic model.

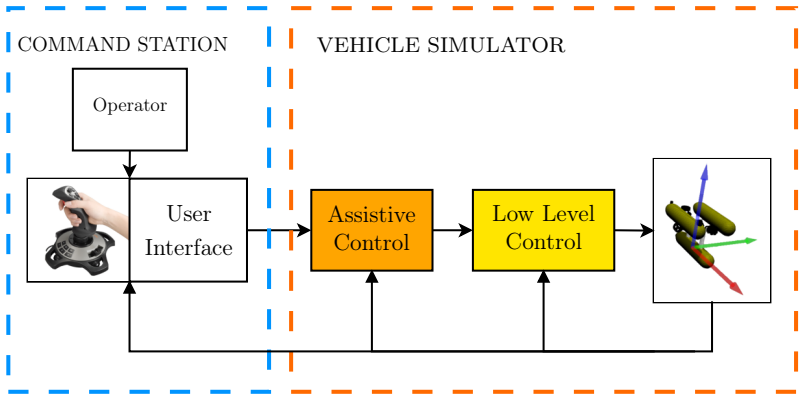

Figure 4.16: Assistive control framework architecture.

The simulations described in the following section have been performed using the Gazebo 3D simulator for the visualization of the ROV motion and the simulation of the needed sensors, while ROS has been used for interfacing it with the control algorithm and the graphical user interface. The Girona 500 CAD model has been used, and it has been equipped with a number of sensors provided by Gazebo:

- **Frontal laser scanner**: a standard planar laser-scanner pointing toward the body-frame x axis of the ROV used for simulating the detection of

potential obstacles that the ROV could encounter during the motion. The obstacles positions are then exploited in the obstacle avoidance task.

- **Bottom laser scanner**: a laser scanner pointing toward the body-fixed z axis exploited for computing the distance from the seabed. This measurement is then rotated in inertial frame and used in the auto-altitude task.
- **Depth-meter**: a simple software node simulating a depth-meter that computes the distance between the ROV and the surface, and that is used to perform the auto-depth task.
- **Cameras**: a front and a rear camera streaming the images on the graphical user interface.

The ROV position and orientation are taken directly from the Gazebo ROS topics, and they are exploited for the position/orientation task, the field of view task and the roll/pitch limit tasks. Figure 4.17 shows a screenshot of the ROV in the simulator where the frontal and bottom laser scanner are visible.

The motion controller has been developed as two nested control loops: the Assistive Control is kinematic and it implements the STPIK algorithm described in section 4.1 giving as output the reference vehicle velocities that fulfill all the tasks, while the Low Level Control computes the torque commands for the motors. The latter has been developed as two interchangeable Gazebo plugins, one kinematic and one dynamic. The kinematic one instantaneously applies the desired velocity coming from the Assistive Control to the ROV in the simulator. It is useful to debug the high-level kinematic control. The dynamic low-level controller is implemented as a standard PID that simulates a second order dynamics on the ROV velocity, making the simulation more realistic.

Figure 4.18 shows the graphical interface that has been developed exploiting Qt, which is a cross-platform application framework used for application software development. The control panel is divided into two sub-block to increase the readability. In detail, the interface displays the two videos taken from the cameras on-board the vehicle and some important telemetry data, such as altitude, depth and number of left/right spins of the ROV. Furthermore the operator can

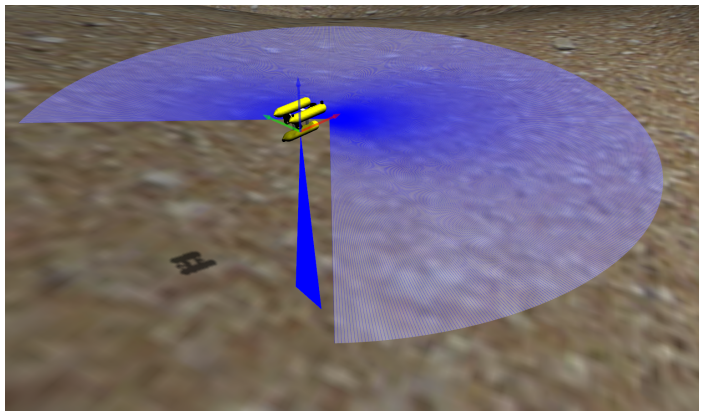

Figure 4.17: Simulation scene: frontal laser scanner along body-fixed frame x axis; bottom laser scanner along the body-fixed frame z axis.

enable/disable a series of tasks: obstacle avoidance (*SB*), auto-attitude (*EB*), auto-heading (*EB*), field of view (*EB*), auto-altitude (*EB* or *SB*), and auto-depth (*EB*), where *EB* and *SB* mean *Equality-Based* and *Set-Based*, respectively. For the auto-depth (*EB*) and the auto-altitude (*EB*) the operator can set the relative desired value; instead, for the auto-altitude (*SB*) he can set the minimum altitude from the seabed which represents the safety distance to prevent damages. It is worth noticing that the task enabling/disabling corresponds to add/remove it to the task hierarchy of the inverse kinematics control changing also the relative priorities. Indeed, the dynamic priority change is managed by the high level controller. The interface gives the operator also the possibility to switch between the inertial and body-frame, for the reasons explained above regarding the command reference frames. Within the objective to help the operator as much as possible, he has the further option to switch only the yaw components between the inertial and body-frame. Finally, the interface presents other two options for the roll and pitch components that allow to set the orientation convention clockwise or counter-clockwise.

4.3.3 ROVs Assistive Framework Simulations

A generic survey mission has been taken into account as simulation case study. In particular, an underwater environment with a non uniform seabed has been

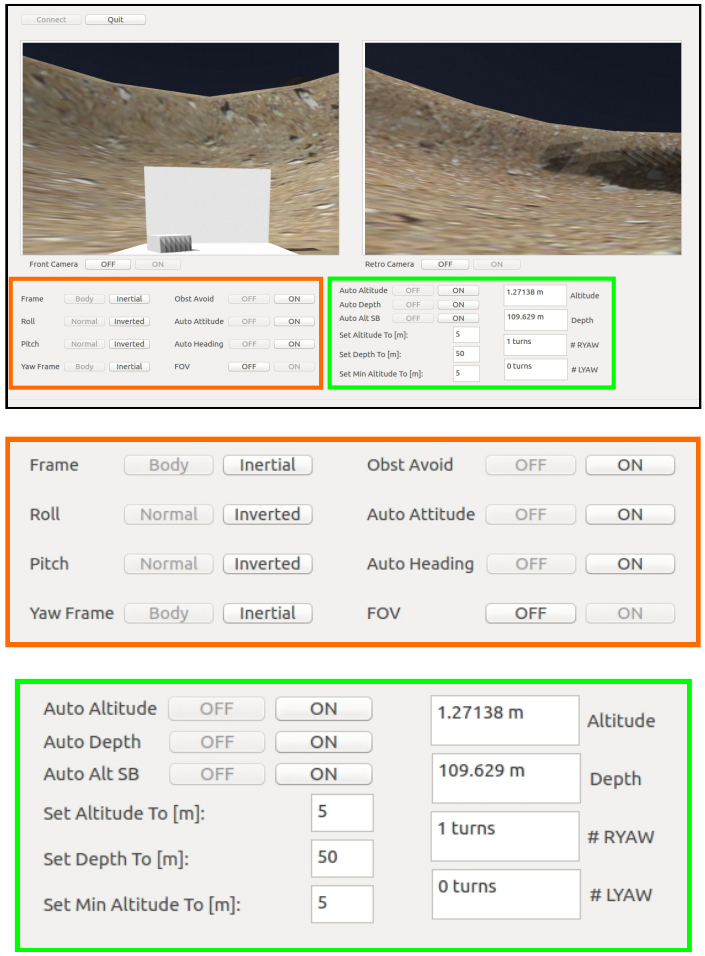

Figure 4.18: Graphical interface: vision panel (two cameras), control panel (divided into two sub-blocks to increase the readability).

rendered adding the presence of a panel.

The validation is made by an operator that drives the vehicle, enabling/disabling the tasks from the graphical interface as explained in the following. In particular, the control framework reduce the complexity of the operation, allowing the operator to focus only on the operational tasks while the safety-related ones are autonomously handled by the system.

The implemented task values and errors have been plotted in real-time during the simulation. In particular, the equality-based task errors and the set-based

task values with the corresponding activation thresholds (black lines) are shown in Fig. 4.19, 4.20 and Fig. 4.21, respectively, where the green and pink plot backgrounds represent the enabled and disabled states of the corresponding task.

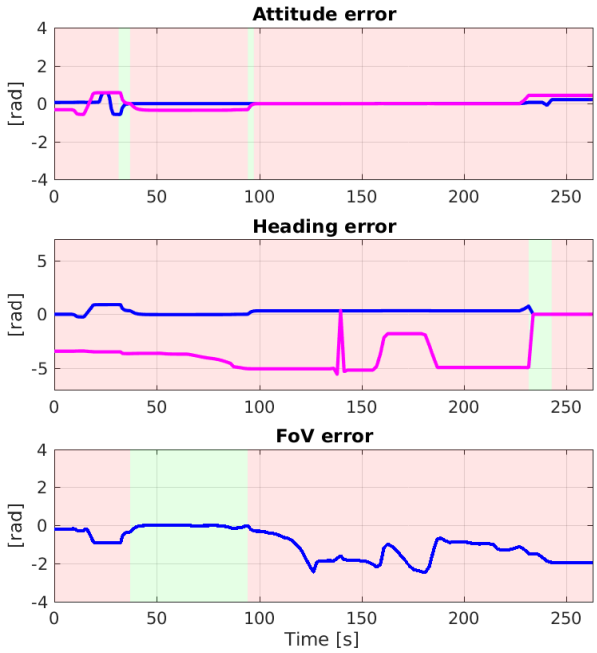

Figure 4.19: Task error plots: attitude ($m = 2$), heading ($m = 2$) and field of view ($m = 1$). The green and pink background represents the enabled and disabled state, respectively, of the corresponding task.

In detail, aiming at testing the entire control framework, all tasks have been enabled/disabled approaching the vehicle to the panel. First of all, a series of rotations have been performed keeping still the vehicle to verify the roll and pitch limits respect ($\sigma_{a,l} = -\pi/4$, $\sigma_{a,u} = \pi/4$ for both ones). This can be

observed during the first 50 seconds in the third and fourth plot in Fig. 4.21. Then the attitude task has been enabled just to set roll and pitch angles to zero (see first plot in Fig. 4.19). With the aim of directing the camera (and therefore also the vehicle) toward the panel, the field of view task has been enabled, as observable in the third plot in Fig. 4.19 in the green range where the error goes to zero. Furthermore, since both the attitude and field of view tasks are fulfilled controlling the angular components their simultaneous enabling does not make sense for guiding the vehicle. This is implemented in the graphical interface in such a way that enabling one implies disabling the other one. At the same time also the obstacle avoidance has been enabled. It becomes active as soon as the vehicle is close to the panel ($\sigma_{a,l} = 2\,\text{m}$). Then the altitude task has been enabled setting the desired value $\sigma_{i,d} = 2\,\text{m}$ as noticeable from the null error in the first plot in Fig. 4.20. Thus the depth task has been enabled with $\sigma_{i,d} = 100\,\text{m}$. Altitude and depth tasks act on the same work space component and therefore, for the same reasons mentioned above, they can not be enabled simultaneously. Then the set-based altitude task has been enabled setting $\sigma_{a,l} = 5\,\text{m}$. In particular, as it is noticeable in the second plot in Fig. 4.21, there are values under the lower threshold. This is due to the intentional decision to move the ROV over the panel, causing a temporary violation of the minimum altitude. Finally the heading task has been enabled to control the pitch and yaw angles, as shown in the second plot in Fig. 4.19. It is worth noticing that spikes present in position and orientation errors, shown in Fig. 4.20 (third and fourth plot), are due to the operator inputs. Indeed he sends position commands that are constant increments with respect to the current ROV pose.

Another aspect taken into consideration during the simulation is the number of twistings. Indeed, within the assumption to have the tether and reading the corresponding values from the graphical interface, the number of clockwise spins has been compensated by the number of counter clockwise ones to prevent twistings.

A video of the simulation with the implemented GUI and the rendered underwater environment is available online at https://youtu.be/yPIaYOFRmwE.

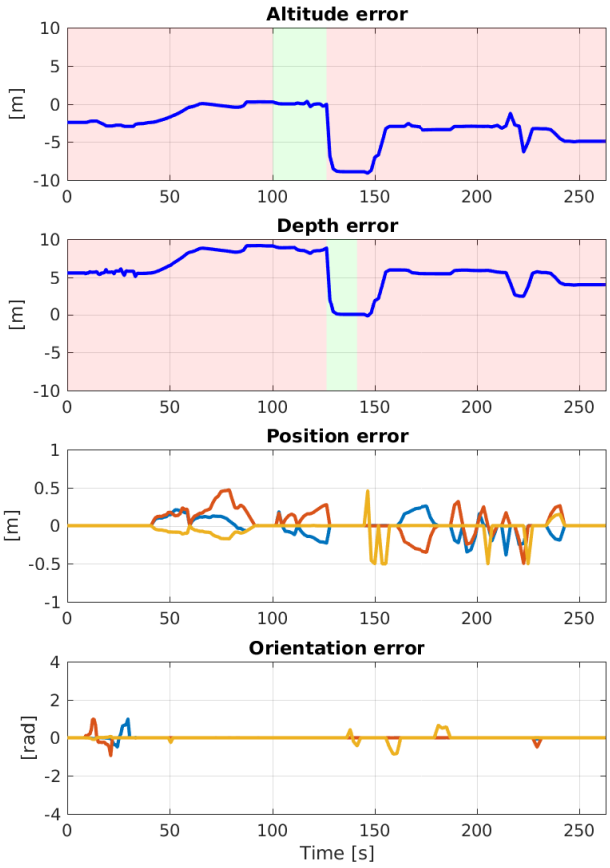

Figure 4.20: Task error plots: altitude ($m = 1$), depth ($m = 1$), position ($m = 3$) and orientation ($m = 3$). The green and pink background represents the enabled and disabled state, respectively, of the corresponding task.

4.3.4 Conclusions

The proposed framework is focused in giving support to a human operator in guiding and manoeuvring a ROV. All the implemented functionalities have been specifically chosen to address real needs that arose after interviews with professional ROV pilots. Simulation results including different task hierarchies

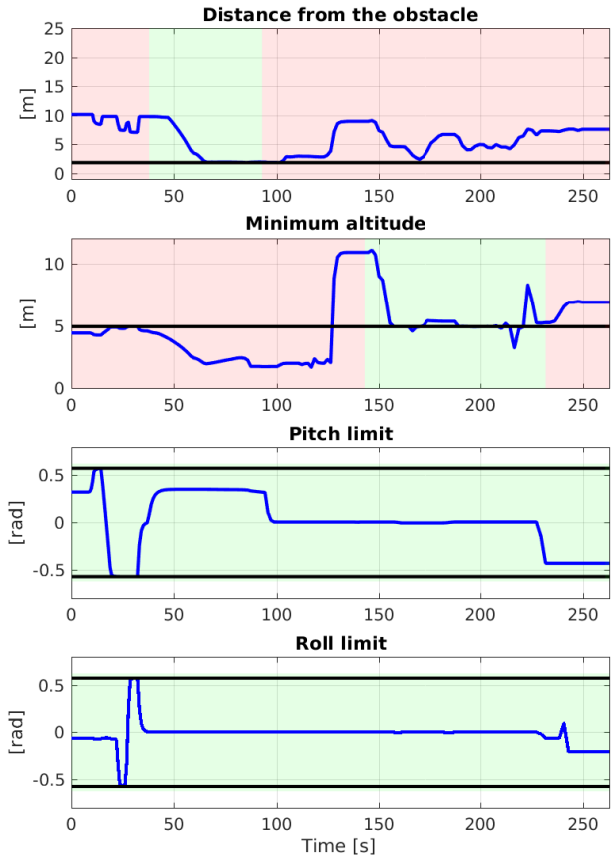

Figure 4.21: Task value plots: obstacle avoidance, auto-altitude, pitch limit and roll limit. The green and pink background represents the enabled and disabled state, respectively, and the black lines are the upper/lower activation thresholds of the corresponding set-based task.

suitable for several kind of operations have been shown, proving the STPIK algorithm effectiveness as well. Future works will be focused on the experimental implementation of the designed control framework.

4.4 The DexROV project

DexROV is an EC (European Commission) Horizon 2020 funded project [2] that aims to develop a system able to perform underwater operations using a novel paradigm that allows the far distance teleoperation of a ROV (Remotely Operated Vehicle) via a satellite communication. This would lead to the usage of a smaller and cheaper support vessel, since a part of the crew would be located in an onshore control center. Satellite communications introduce a non-negligible delay that has to be properly handled by the system in order to effectively perform the needed operations. The latency mitigation strategy includes a simulation environment and a cognitive engine. The operator interacts with the ROV in the simulation environment that receives 3D data from the perception system, performing the desired movements with a force-feedback exoskeleton without taking into account time latencies and instructing a cognitive engine that generates motion and manipulation primitives to be sent to the real ROV. Figure 4.22 represents the project's concept.

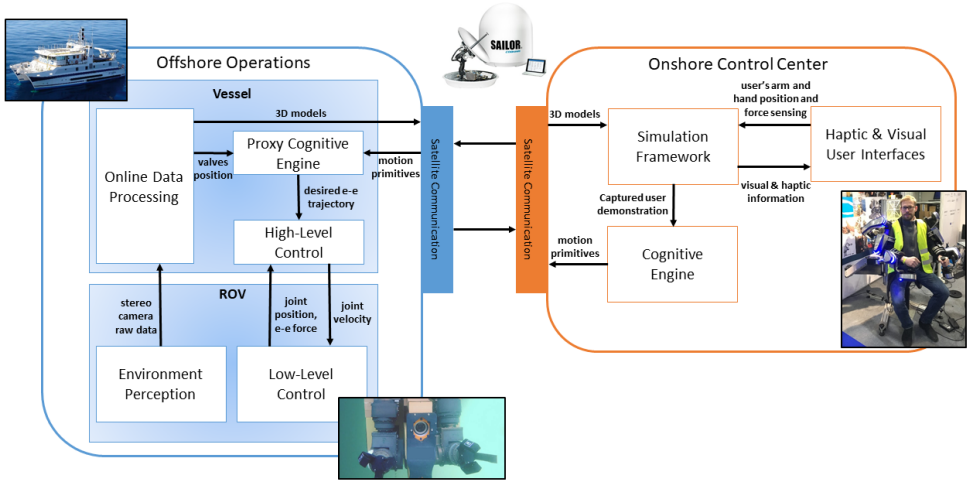

Figure 4.22: DexROV project concept.

The perception system performs 3D data acquisition through a stereo camera, doing processing of the needed information and sending them to the control

center in real-time [15]. Furthermore the ROV is equipped with an AHRS (Attitude and Heading reference System), a DVL (Doppler velocity log) and a USBL (ultra-short baseline) that are concurrently used for its accurate pose estimation [59]. The cognitive engine is split in two parts: on the onshore side it recognizes the actions that the operator wants to perform learning from demonstrations; on the offshore side it reconstructs the motion primitive despite of the non homogeneous communication latency. This is achieved by exploiting a task parametrized Gaussian Mixture Model that adapts the reference end-effector trajectory to the dynamic environment in which the ROV operates [18].

4.4.1 Vehicle-Manipulator System

The DexROV vehicle-manipulator system is composed by a commercial medium-class 4 DOFs (x, y, z, yaw) ROV, the Sub-Atlantic Apache 2500, and a custom skid mounted on the lower part carrying on the vision system and two arms, a 6 DOFs manipulator and a 3 DOFs clamping arm, as shown in Fig 4.23. In

Figure 4.23: The DexROV system on board the Janus II vessel.

particular, the clamping arm allows to perform operations that require more strength such as Oil & Gas industry related tasks. However, a structure to clamp at is necessary. In the specific case, aimed at testing the system manipulation capabilities, a mock-up panel has been designed and built, shown in

Fig. 4.24, including a set of standard ISO interfaces such as valves and connectors commonly used in the Oil & Gas industry.

Figure 4.24: Mock-up panel designed for the trials on the left; flat valves detail on the right.

4.4.2 Control Architecture

The overall control architecture is shown in Fig. 4.25. As noticeable, it is composed by three nested loops:

- *admittance filter.* It is the outer loop designed to handle the interaction between the end-effector and the panel as well as unexpected collisions caused by eventual errors in the valve pose estimation or by external disturbance such as the ocean current. It takes as input the requested trajectory $\boldsymbol{p}_{\text{req}}$ from the cognitive engine and the force/torque measurements coming from the wrench sensor placed on the wrist of the manipulator. Then, it computes the desired trajectory $\boldsymbol{p}_{\text{des}}$ for the set-based inverse kinematics algorithm. In particular, $\boldsymbol{p}_{\text{des}}$ is computed in order to make the manipulator compliant with respect to the external forces.
- *set-based inverse kinematics algorithm.* Taking as input the desired trajectory, it computes the reference velocities $\boldsymbol{\zeta}_{\text{des}}$ for the vehicle-manipulator system performing simultaneously safety related tasks as well.
- *vehicle-arm dynamic controller.* It applies the forces/moments and joint torques necessary to make the system follow the input reference velocities.

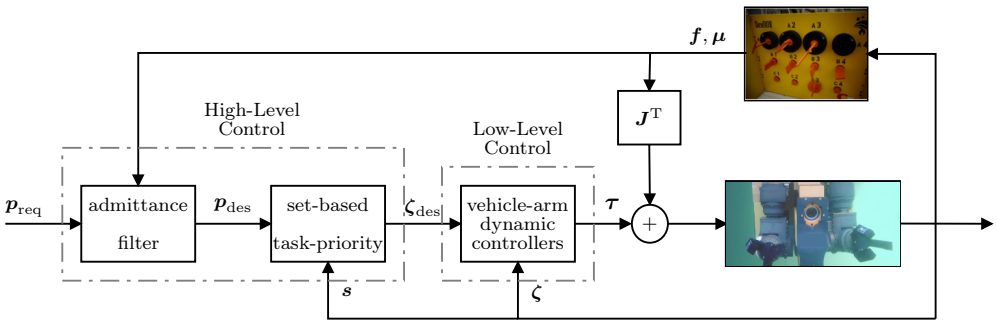

Figure 4.25: Implemented control architecture.

4.4.3 Implemented Control Tasks

Aimed at guaranteeing the DexROV system safety, the following set-based tasks have been implemented:

- **Joint Limits**. Such a task can be expressed by the following inequality for each one of the joints:

$$q_{i,m} \leq q_i \leq q_{i,M} \ , \tag{4.31}$$

  where q_m and q_M are the minimum and the maximum position that the joint will reach. The task Jacobian is a row vector filled with zeros and a 1 only at the $i-$th position:

$$\boldsymbol{J}_{jl,i} = \begin{bmatrix} \mathbf{0} & \cdots & \underbrace{1}_{i} & \cdots & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{1\times n} \ . \tag{4.32}$$

- **Arm Manipulability**. It consists of keeping the manipulator far from singular configurations that would have undesirable effects on the joint velocity computation. The control objective can be expressed as:

$$w \geq w_m \tag{4.33}$$

  where w is the measure of manipulability defined as [76]:

$$w = \sqrt{\det(\boldsymbol{J}_{\text{arm}}\boldsymbol{J}_{\text{arm}}^{\text{T}})} \ , \tag{4.34}$$

where

$$\boldsymbol{J}_{\text{arm}} = \begin{bmatrix} \boldsymbol{J}_{\text{pos,arm}} \\ \boldsymbol{J}_{\text{ori,arm}} \end{bmatrix} \tag{4.35}$$

is the matrix stacking the arm position and orientation Jacobian matrices. The manipulability task Jacobian $\boldsymbol{J}_w$ is computed numerically following the procedure outlined in [49].

- **Virtual Wall**. This task makes the end-effector stay always above a minimum distance from a virtual plane has been implemented, in order to avoid collisions between the arm and the vehicle itself. The constraint can be expressed as:

$$d \geq d_m \tag{4.36}$$

where d_m is the desired minimum distance between the end-effector and the plane and:

$$d = \hat{\boldsymbol{n}}^{\mathrm{T}}(\boldsymbol{p}_e - \boldsymbol{p}_1) \,, \tag{4.37}$$

where $\boldsymbol{p}_e$ is the end-effector position and $\hat{\boldsymbol{n}}$ is the outer normal unit vector from the plane computed as:

$$\hat{\boldsymbol{n}} = \frac{(\boldsymbol{p}_2 - \boldsymbol{p}_1) \times (\boldsymbol{p}_3 - \boldsymbol{p}_1)}{||(\boldsymbol{p}_2 - \boldsymbol{p}_1) \times (\boldsymbol{p}_3 - \boldsymbol{p}_1)||} \tag{4.38}$$

and $\boldsymbol{p}_1$, $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$ are three points belonging to the plane. The task Jacobian is computed as:

$$\boldsymbol{J}_{vp} = -\hat{\boldsymbol{n}}^{\mathrm{T}} \boldsymbol{J}^0_{\text{pos,arm}} \,, \tag{4.39}$$

where $\boldsymbol{J}_{\text{pos}}$ is the position Jacobian matrix.

The task priority order has been assigned taking into consideration the following three categories with decreasing priority [22]:

1. **Safety tasks:** tasks such as mechanical joint limits and virtual walls are necessary to assure the safety of the system, thus the highest priority level needs to be assigned to them;

2. **Operational tasks:** the end-effector pose and configuration tasks are necessary to perform the desired operation and they have to be executed

in the null space of the safety tasks;

3. **Optimization tasks:** the manipulability task can be seen as an optimization one, in which the objective is to maximize the manipulability measure while performing all the higher-priority tasks. Nevertheless, it can be used as a safety task as well, setting a minimum threshold that the system does not have to exceed during the movement.

4.4.4 Admittance Control

An admittance control node has been implemented in order to make the system compliant with respect to undesired external forces. Indeed unexpected collisions could easily damage the panel or the arm itself, given the dimension and the weight of the vehicle.

More in detail, the admittance control allows to design a specific dynamic behaviour of the system in case of contact with the environment by setting three parameters representing the inertia $\boldsymbol{K}_m$, the damping $\boldsymbol{K}_d$ and the stiffness $\boldsymbol{K}_k$ of a virtual mass-spring-damper mechanical system. The software control node takes as input the requested position/quaternion $\boldsymbol{x}_r$, linear/angular velocity $\dot{\boldsymbol{x}}_r$ and linear/angular acceleration $\ddot{\boldsymbol{x}}_r$ for the end-effector coming from the proxy cognitive engine and the measurements of the force/torque sensor and gives as output a new desired trajectory $(\boldsymbol{x}_d, \dot{\boldsymbol{x}}_d, \ddot{\boldsymbol{x}}_d)$ that complies with the desired mechanical impedance.

The acceleration output by the admittance filter is:

$$\ddot{\boldsymbol{x}}_d = \boldsymbol{K}_m^{-1} \left[\boldsymbol{K}_k \tilde{\boldsymbol{x}} + \boldsymbol{K}_d \dot{\tilde{\boldsymbol{x}}} + \boldsymbol{K}_m \ddot{\boldsymbol{x}}_r + \boldsymbol{h}^0_{\text{ext,sel}} \right] , \tag{4.40}$$

where $\tilde{\boldsymbol{x}} = \boldsymbol{x}_r - \boldsymbol{x} \in \mathbb{R}^6$ is the vector stacking the position and the quaternion error, $\dot{\tilde{\boldsymbol{x}}} = \dot{\boldsymbol{x}}_r - \dot{\boldsymbol{x}} \in \mathbb{R}^6$ is the vector stacking the linear and angular velocity error and $\boldsymbol{h}^0_{\text{ext}} \in \mathbb{R}^6$ is defined as:

$$\boldsymbol{h}^0_{\text{ext,sel}} = \boldsymbol{K}_f \boldsymbol{h}^0_{\text{ext}} , \tag{4.41}$$

where $\boldsymbol{h}_{\text{ext}}^{0}$ are the measured external forces and torques expressed in the arm base frame, and $\boldsymbol{K}_f$ is a selection diagonal matrix of 0 and 1 that is used to decide whether or not the system has to be compliant in a certain direction. The desired linear/angular velocity $\dot{\boldsymbol{x}}_d$ and position/quaternion $\boldsymbol{x}_d$ are then computed by numerical integration of the acceleration.

4.4.5 First Experimental Campaign

A first experimental campaign has been conducted in June 2017 with the on-shore control center located in Brussels (Belgium) and the offshore operations performed in Marseilles (France).

Several tests have been executed with the system in different configurations, accepting the end-effector trajectories by different means, i.e., by local code or joystick, by remote code or joystick and finally by remote exoskeleton. In the remote configuration, the trajectory is generated in Brussels (Belgium) and then transmitted via satellite communications to the vessel in Marseilles (France) and then through the umbilical to the vehicle. In particular, regarding the satellite communication, the nominal data bandwidth for the uplink from the vessel was 768 Kb/s and the downlink to the vessel was 256 Kb/s, with a nominal round-trip delay of 620 ms [78].

In the following two experiments are reported:

- *Experiment 1.* A position and orientation task is performed taking into account a singular configuration;
- *Experiment 2.* A position and orientation task is performed fulfilling joint limits.

Position and orientation, singular configuration

In the first test the end-effector position and orientation task is given. The desired trajectory is a simple circle on the y-z plane in the arm base frame at a constant velocity, while keeping the orientation at a constant value. It is worth noticing that the manipulator intentionally reaches a singular configuration

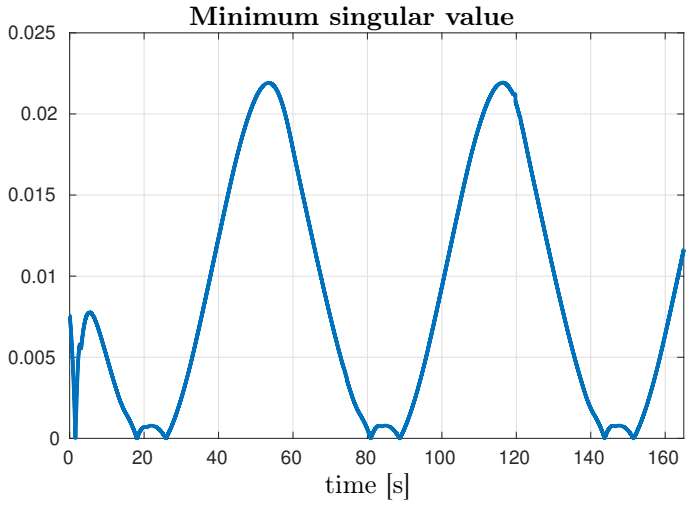

Figure 4.26: First experiment - position and orientation control: minimum singular value of $\boldsymbol{J}$ over time. The arm intentionally reaches a singular configuration during the trajectory.

during the trajectory, as the minimum singular value reaches very small values. Figure 4.26 shows the minimum singular value of the $\boldsymbol{J}$ matrix over time.

Figures 4.27 and 4.28 show the position and orientation error together with the joint positions during the experiment. The DLS pseudoinverse prevents the chattering phenomenon on the joint velocities, generating a higher error on the orientation while the position error remains sufficiently low during the whole trajectory.

Mechanical joint limits

In the second experiment the system is asked to follow the same circular trajectory without controlling the orientation while keeping the fifth joint below a certain threshold. The prioritized task hierarchy imposed is:

1. Joint 5 maximum threshold
2. End-effector position

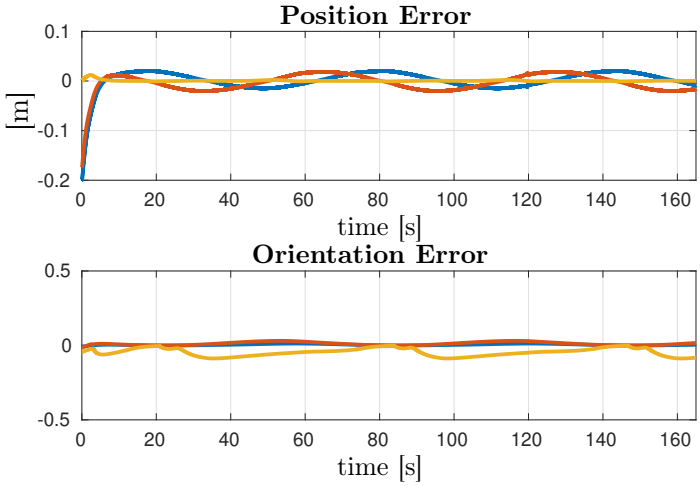

Figure 4.27: First experiment - position and orientation control: position and orientation error over time; the position error is kept low during the entire trajectory, while the orientation error grows for the effect of the joint velocities damping.

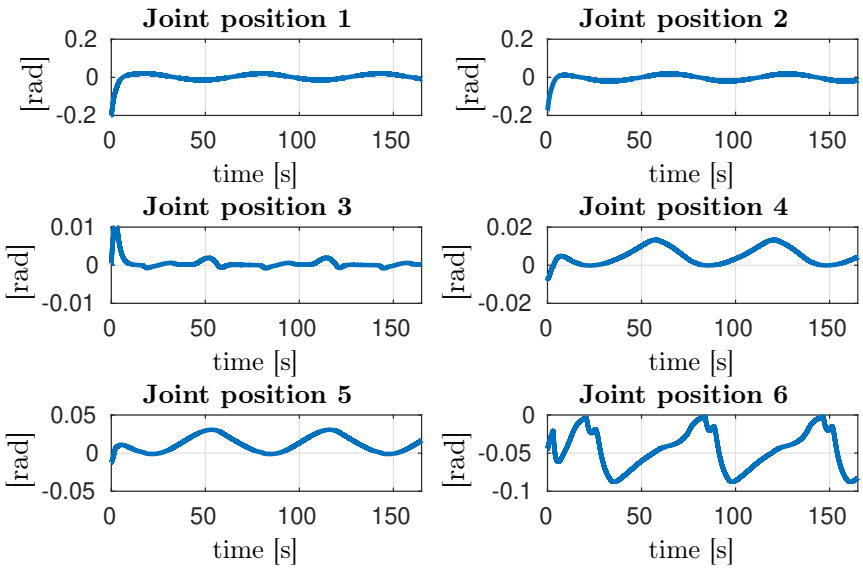

Figure 4.28: First experiment - position and orientation control: joint positions over time.

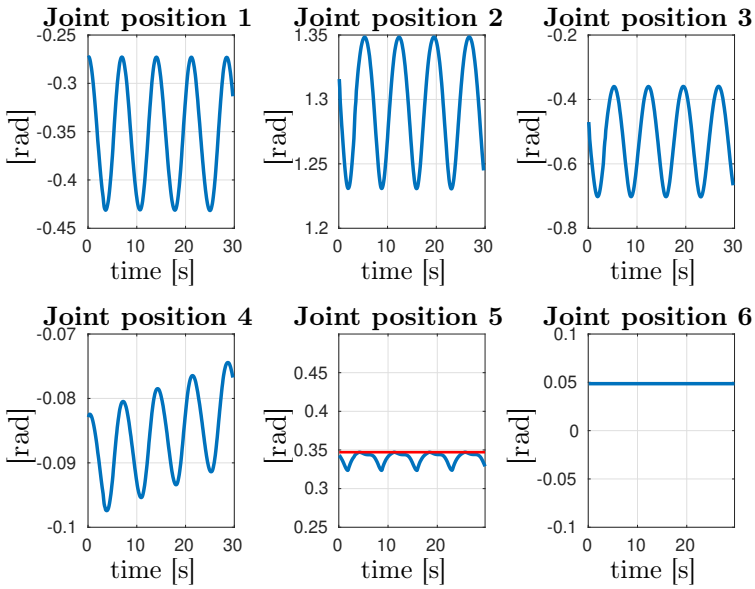

Figure 4.29: Second experiment - 5th joint mechanical limit and position control: joint positions and upper threshold on the fifth joint (in red). The fifth joint position remains always below the chosen threshold.

Figures 4.29 and 4.30 show the position error and the joint values during the experiment. The set-based inverse kinematics algorithm makes the joint position stay below the chosen threshold (in red), while the trajectory is followed with a low position error.

Then another joint limit has been added as control objective, giving the following hierarchy:

1. Joint 3 maximum threshold
2. Joint 5 minimum threshold
3. End-effector position

Figure 4.31 shows the joint positions during the experiment, while Fig. 4.32 shows the position error. It is worth noticing that the third joint starting position is above the chosen maximum threshold, but the control algorithm quickly bring its value to the imposed limit. From that point, both the joint

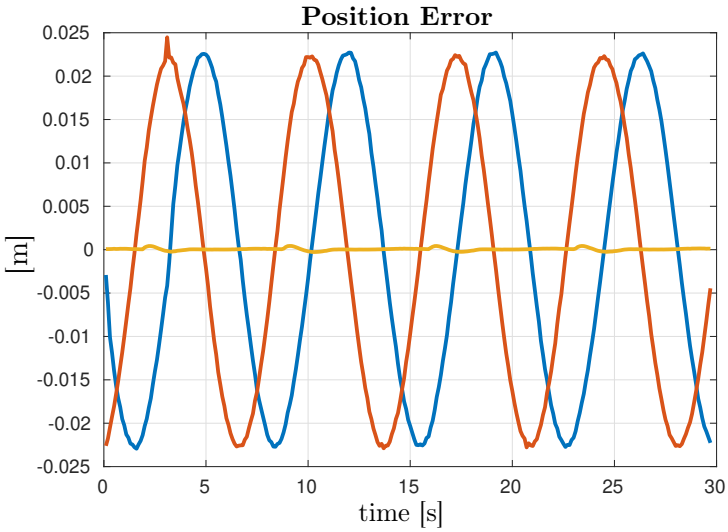

Figure 4.30: Second experiment - 5th joint mechanical limit and position control: position error over time.

limits are satisfied during the entire trajectory. The position error grows with respect to the other experiment because the combination of the third and fifth joint mechanical limits, being at a higher priority level with respect to the position task, reduces the end-effector operational workspace.

4.4.6 Second Experimental Campaign

A second experimental campaign has been conducted in June 2018. As for the first campaign, the onshore control center was located in Brussels (Belgium) and the offshore operations has been performed in Marseilles (France) at a 30 m depth. The experiments have involved the entire DexROV pipeline in a "turn the valve" operation: on the onshore side the operator has performed the valve turning with the exoskeleton in the virtual reality environment, monitored by the cognitive engine. On the offshore side the proxy cognitive engine has generated the reference trajectory for the end-effector [38], while the vision system was in charge of the valve poses estimation.

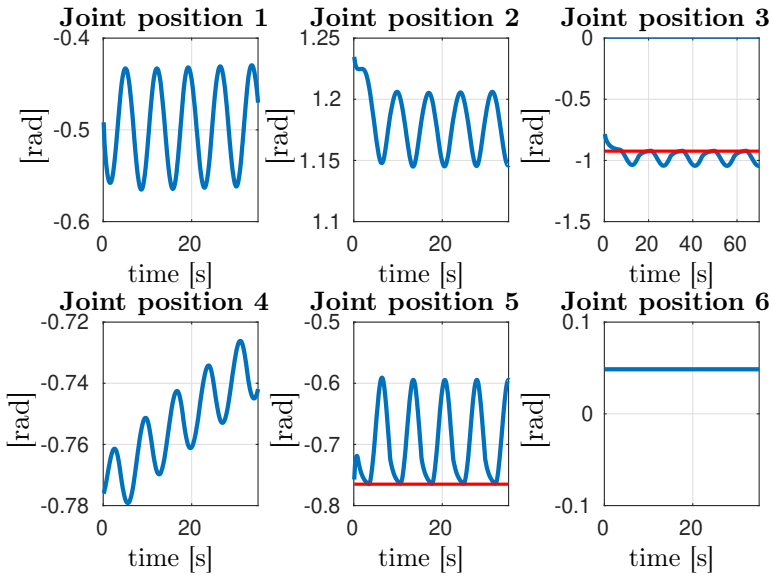

Figure 4.31: Second experiment - 3rd and 5th joints mechanical limit and position control: joint positions and minimum/maximum thresholds (in red).

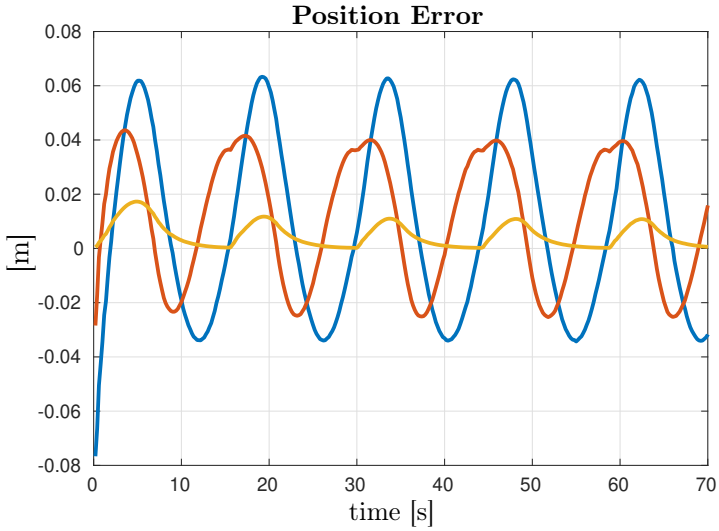

Figure 4.32: Second experiment - 3rd and 5th joints mechanical limit and position control: position error over time. Notice that the error is large due to the constraints and intentional low feedback gains.

From the admittance filter perspective, the "turn the valve" operation can be divided into two phases: the "approach" and "rotating" phase, respectively. During the "approach" phase the selection matrix $\boldsymbol{K}_f$, defined in Eq. (4.41), is set as the identity matrix, making the system compliant in all the directions and angles. When the operation state switches to the "rotating" phase, the proxy cognitive engine automatically triggers a signal to the admittance controller. Then, the last element on the diagonal of the selection matrix is changed to 0, making the system stiff on the z angle and, therefore, allowing the valve rotation while remaining compliant on the other directions.

The task hierarchy implemented for the trials was composed by 7 tasks:

1. set-based: joint limits for each actuator (6 tasks 1-dimensional each)
2. equality-based: end-effector configuration, (1 task 6-dimensional)

The joint limits have been chosen in order to match their actual mechanical limits and to avoid collisions with the cameras and the buoyancy foam placed between the two arms. It is worth noticing that, due to the fixed-based configuration and the 6 DOF arm structure, other equality-based tasks cannot be achieved. Furthermore, due to the hard timing constraints during the operations at sea, it was not possible to carefully design virtual walls as additional set-based tasks. However, these constraints have been taken into account by properly designing more restrictive joint limits.

The manipulability task, even if implemented, has not been taken into account since, during the "turn the valve" operation, the system is forced to operate in configurations close to singularity due to the 6 DOFs kinematic structure and the space constraints induced by the buoyancy foam. Thus, in such situation the manipulability task would be active for most of the time preventing the operation accomplishment.

In the following, the results of two experiments are shown:

- *Experiment 1.* A "turn the valve" operation, in which the vision system estimated almost perfectly the valve poses, preventing any undesired interaction with the panel, has been performed;

- *Experiment 2.* A "turn the valve" operation has been performed, however, differently form the previous test, the visibility of the scene was not perfect, generating a wrong valve pose estimation and, therefore, causing unexpected collisions.

Turn the valve operation without unexpected collisions

In the first experiment a "turn the valve" operation has been performed in the condition that the vision system almost perfectly estimated the valve position. Indeed, as observable from Fig. 4.33, the force and torque measured by the wrench sensor are both very low. Then, it means that the admittance filter has no effects on the requested trajectory (see Fig. 4.34).

More in detail, the entire operation is split into four steps:

- **step1** - The end-effector reaches a pre-grasp configuration aligned with the valve at around $t = 60\,\text{s}$.
- **step2** - At $t = 90\,\text{s}$ the "rotating" phase starts, turning the valve 90 degrees clockwise.
- **step3** - After turning 90 degrees clockwise, the same amount is performed anti-clockwise.
- **step4** - The end-effector disengages the valve and reaches a rest position.

During these steps the set inverse kinematics controller in addition to the end-effector configuration task (see Fig. 4.35, and Fig. 4.36), fulfills the constraints on all the six joints, as shown in Fig. 4.37.

A video of this experiment taken on board the support vessel is available online at https://youtu.be/6vU6qcOmTKM.

Turn the valve operation with unexpected collisions

In the second experiment a "turn the valve" operation has been performed again. However, differently from the previous experiment, a wrong estimation of the valve panel by the vision system due to poor scene visibility has been taken into

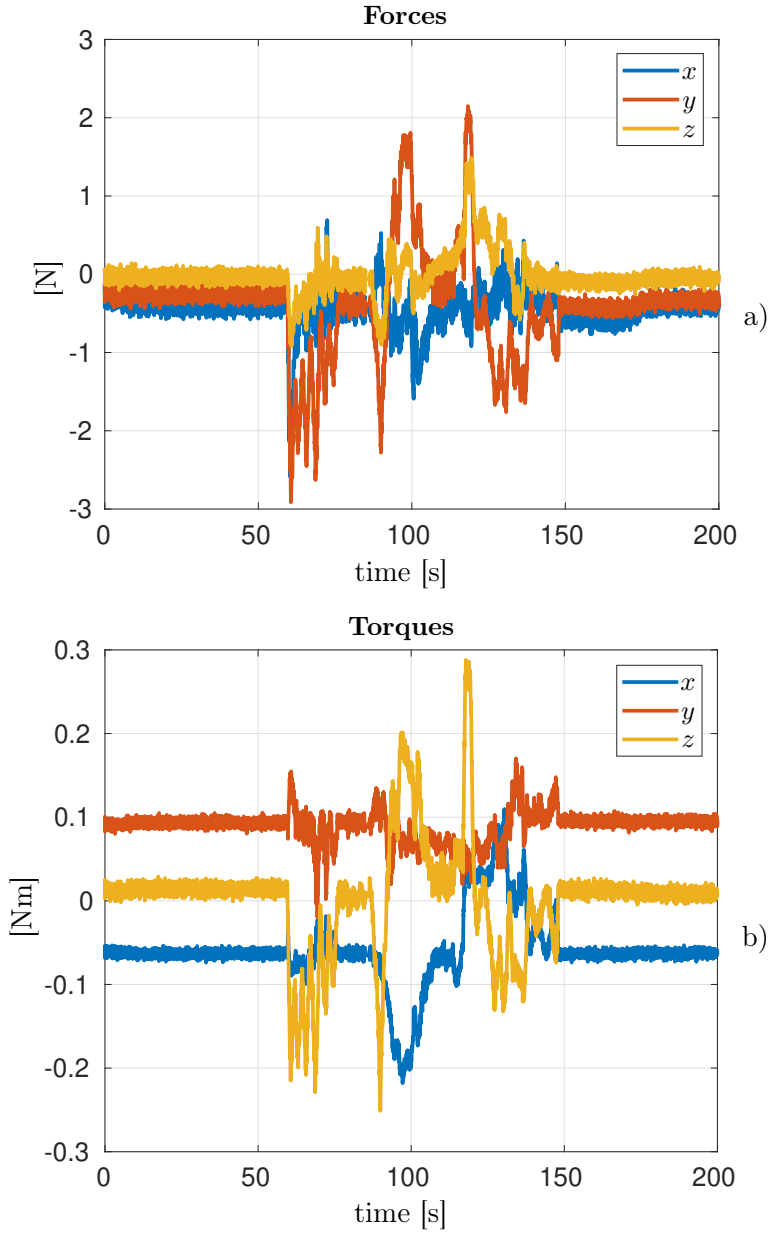

Figure 4.33: Turn valve without unexpected collisions: a) measured force; b) measured torque.

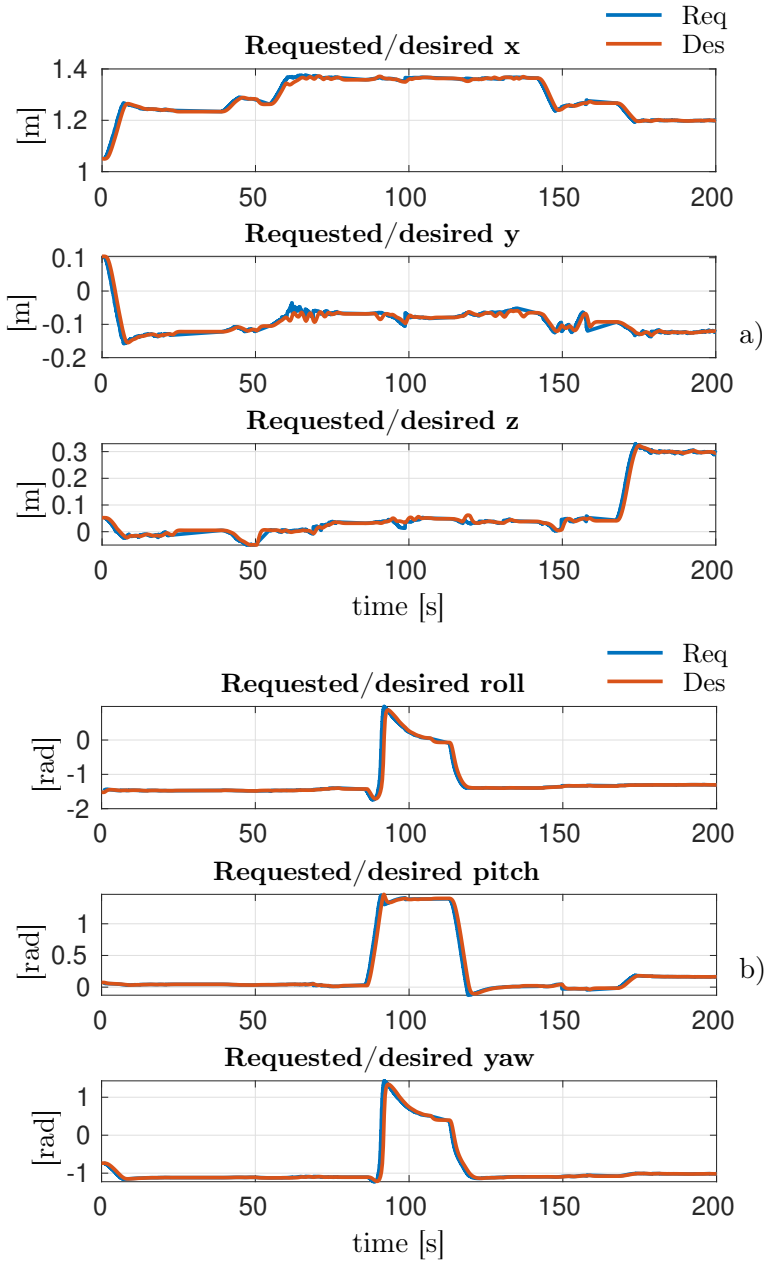

Figure 4.34: Turn valve without unexpected collisions: a) requested (blue) and desired (red) x, y and z coordinates; b) requested (blue) and desired (red) roll, pitch and yaw angles.

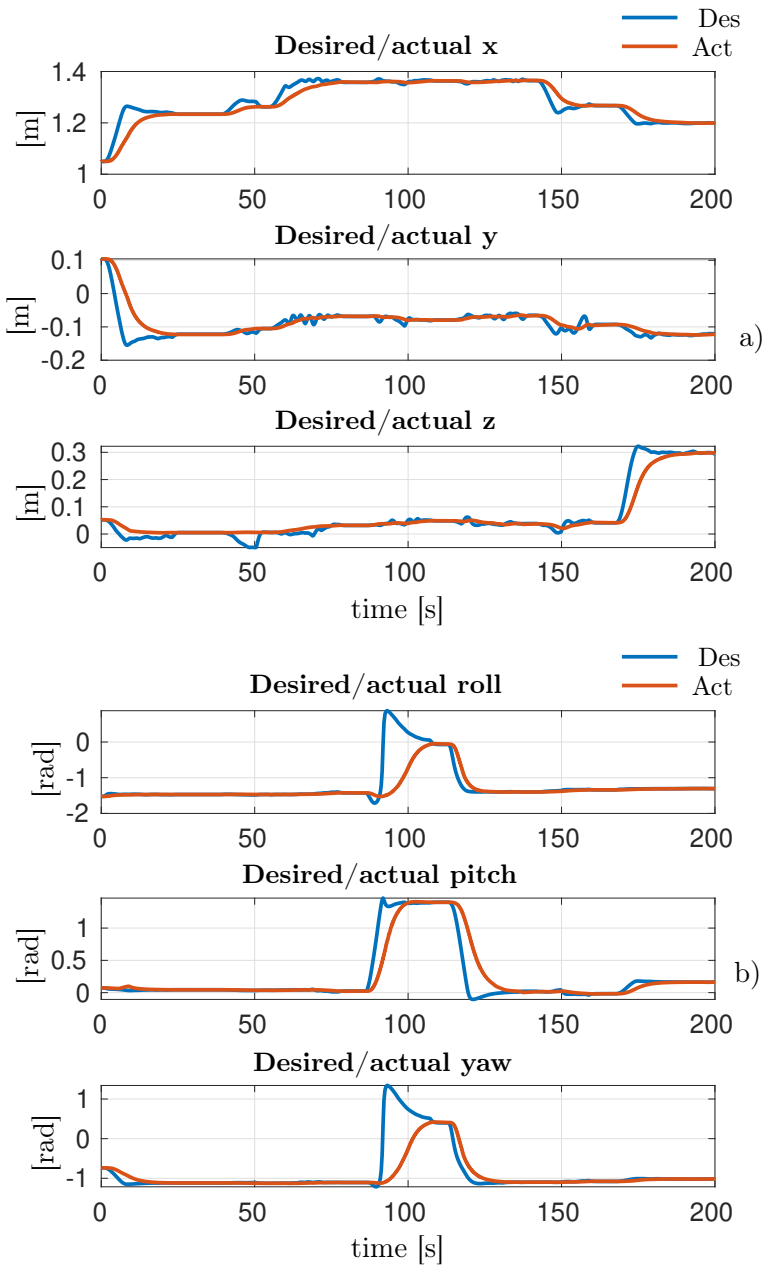

Figure 4.35: Turn valve without unexpected collisions: a) desired (blue) and actual (red) x, y and z coordinates; b) desired (blue) and actual (red) roll, pitch and yaw angles.

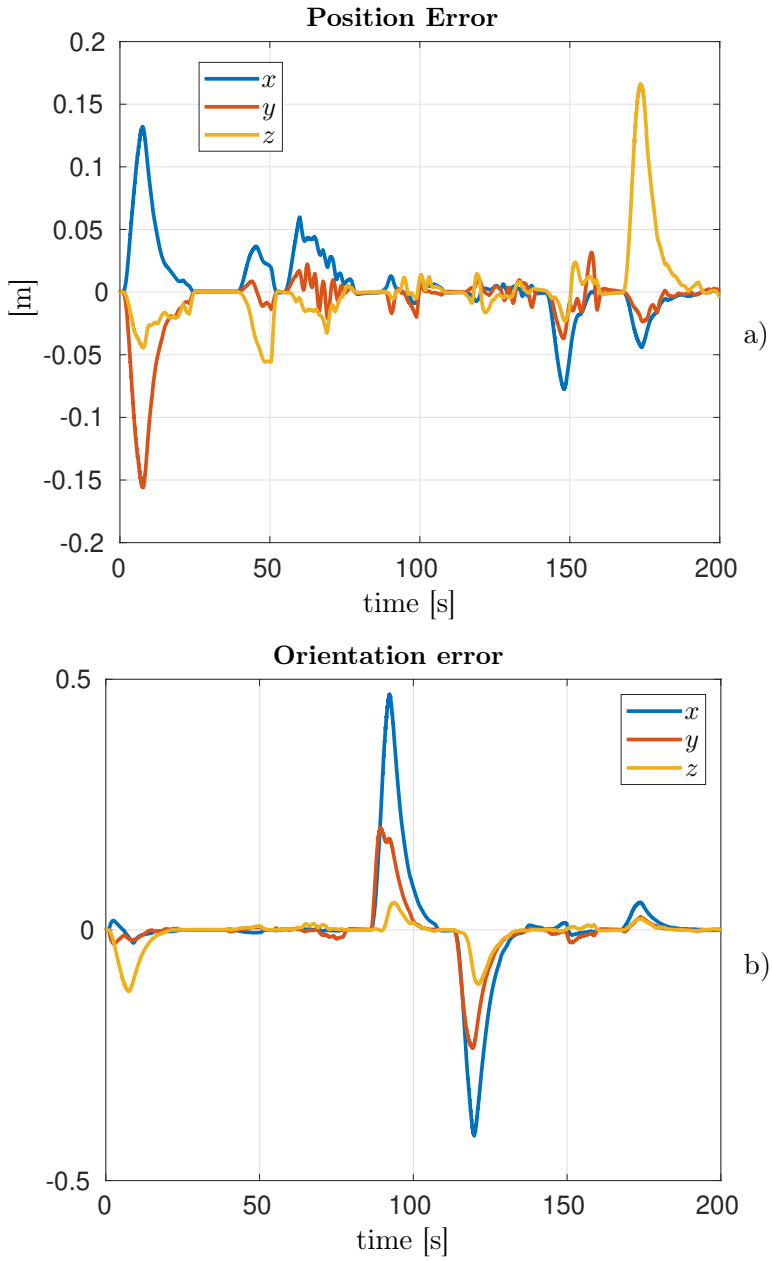

Figure 4.36: Turn valve without unexpected collisions: a) position error over time; b) quaternion error over time.

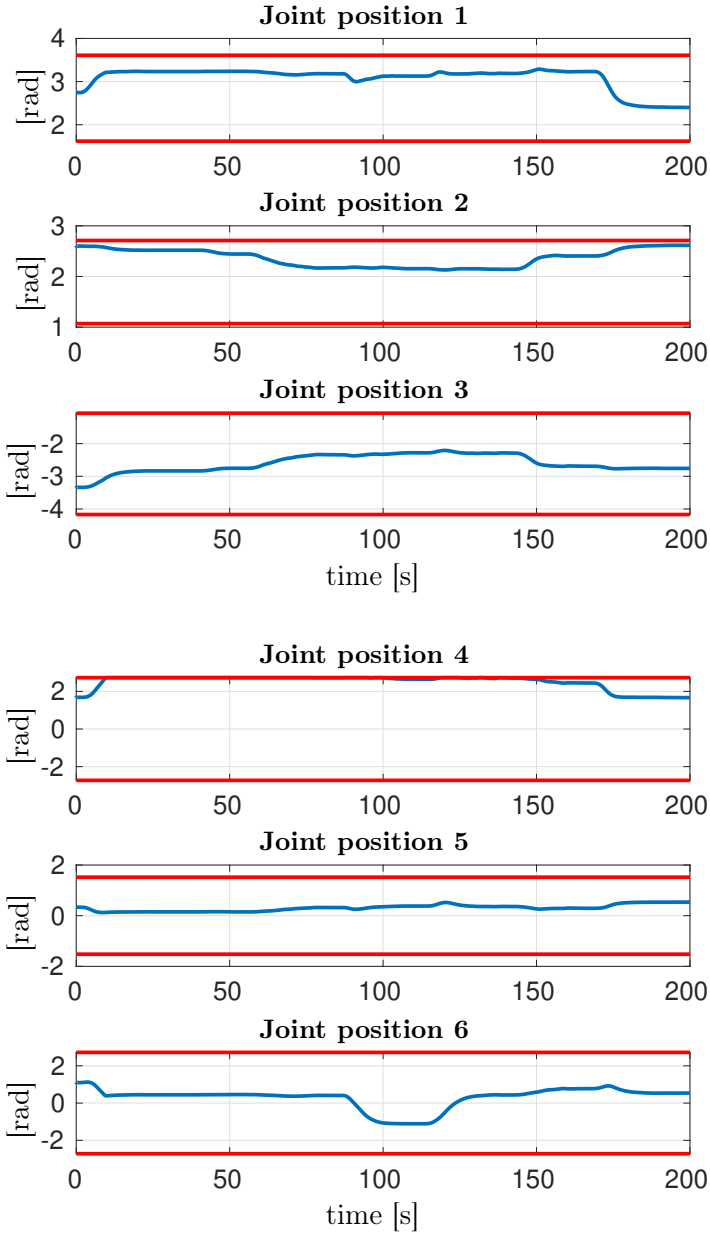

Figure 4.37: Turn valve without unexpected collisions: joint position over time (blue) and imposed thresholds (red).

consideration. Indeed, as shown in Fig. 4.38.a, at $t = 240$s the end-effector hits unexpectedly the panel reaching a sensed force of 12N and torque of 1.5Nm. Nevertheless, the admittance filter successfully handles the unexpected collision modifying the requested trajectory (see Fig. 4.39).

The desired trajectory and the actual end-effector trajectory are reported in Fig. 4.40, while the position and orientation errors are reported in Fig. 4.41. As in the previous experiments the mission is divided in four steps. Thus, the end-effector has first to approach the target valve and to turn it 90 degrees clockwise and then again by the same value anticlockwise going finally to the rest position. During these steps the joint limits are fulfilled as shown in Fig. 4.42.

4.4.7 Conclusions

The DexROV project is another proof of the effectiveness of the Set-based Task-Priority Inverse Kinematics (STPIK) algorithm. Indeed, it has been successfully applied to the DexROV vehicle-manipulator system to perform a "turn the valve" operation, composing the task hierarchy with several safety-tasks in addition to the end-effector configuration. Furthermore, it has been also combined with an admittance control within the aim to manage unexpected collisions between the end-effector and the mock-up panel.

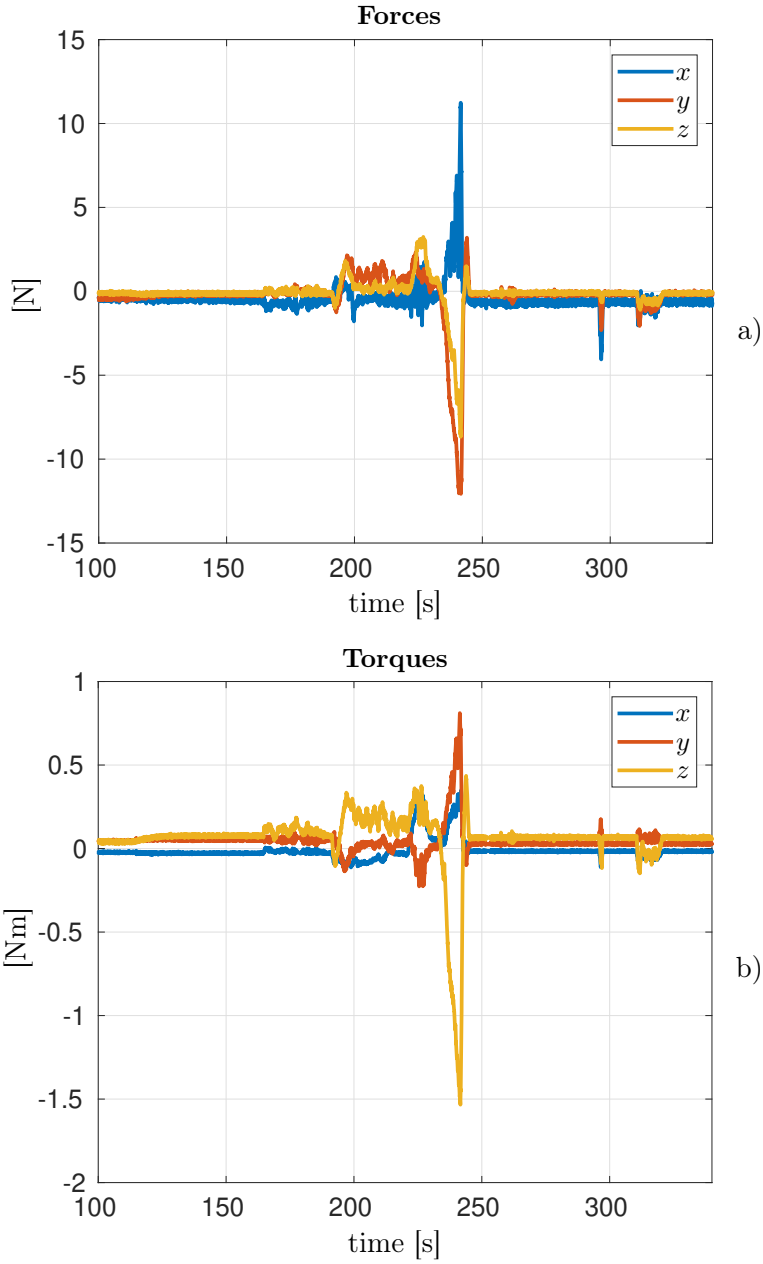

Figure 4.38: Turn valve with unexpected collisions: a) measured force; b) measured torque.

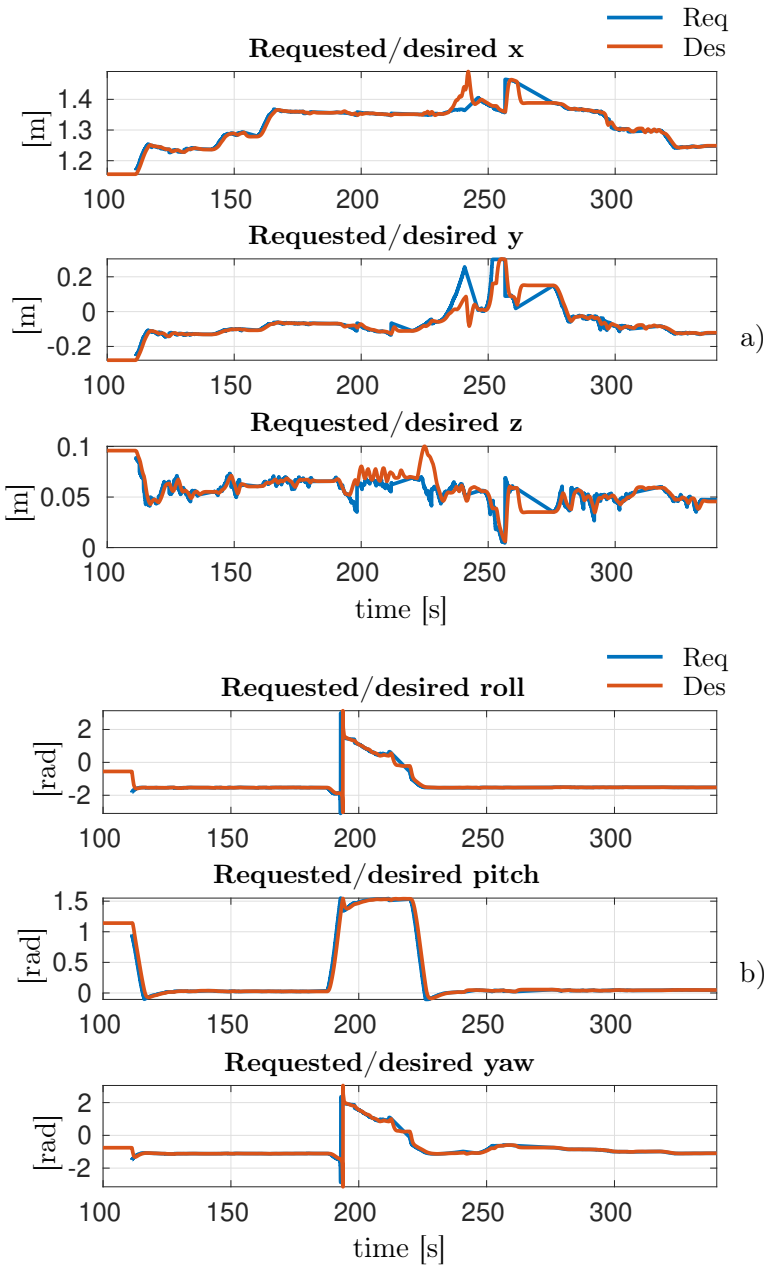

Figure 4.39: Turn valve with unexpected collisions: c) requested (blue) and desired (red) x, y and z coordinates; d) requested (blue) and desired (red) roll, pitch and yaw angles.

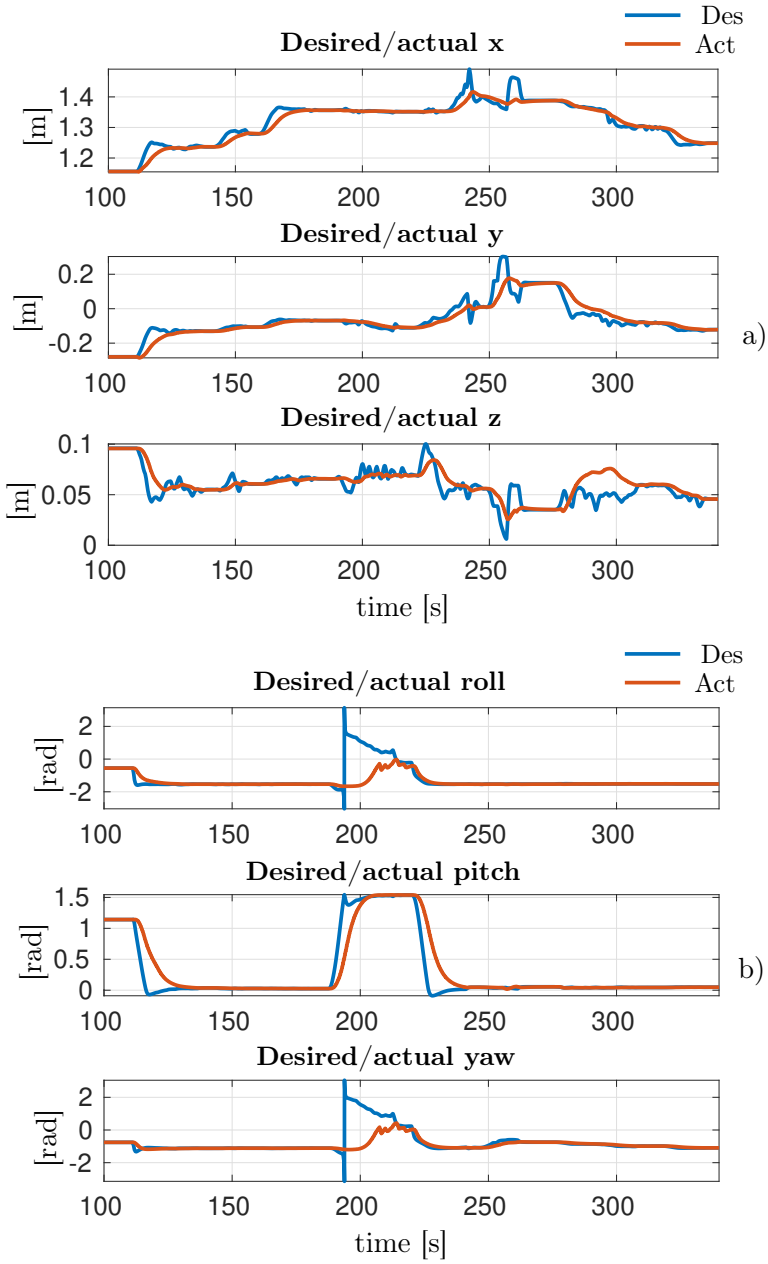

Figure 4.40: Turn valve with unexpected collisions: a) desired (blue) and actual (red) x, y and z coordinates; b) desired (blue) and actual (red) roll, pitch and yaw angles.

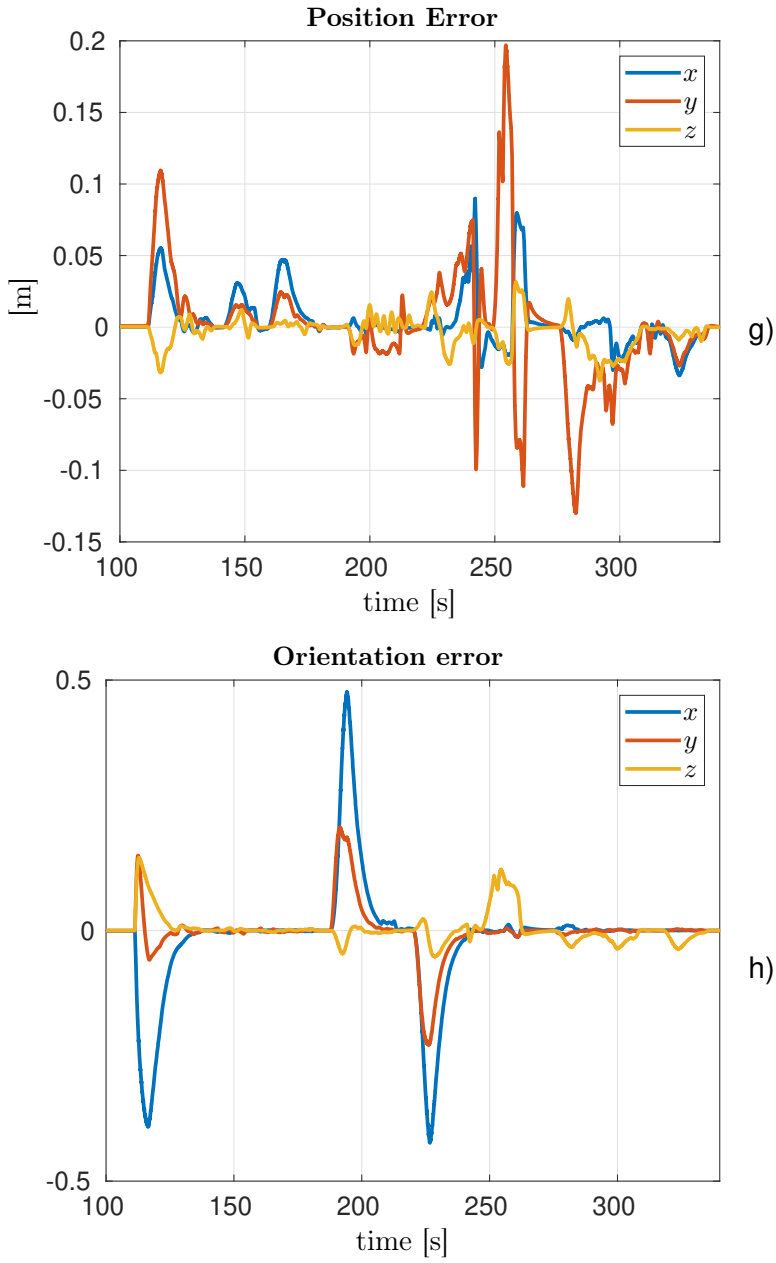

Figure 4.41: Turn valve with unexpected collisions: a) position error over time; b) quaternion error over time.

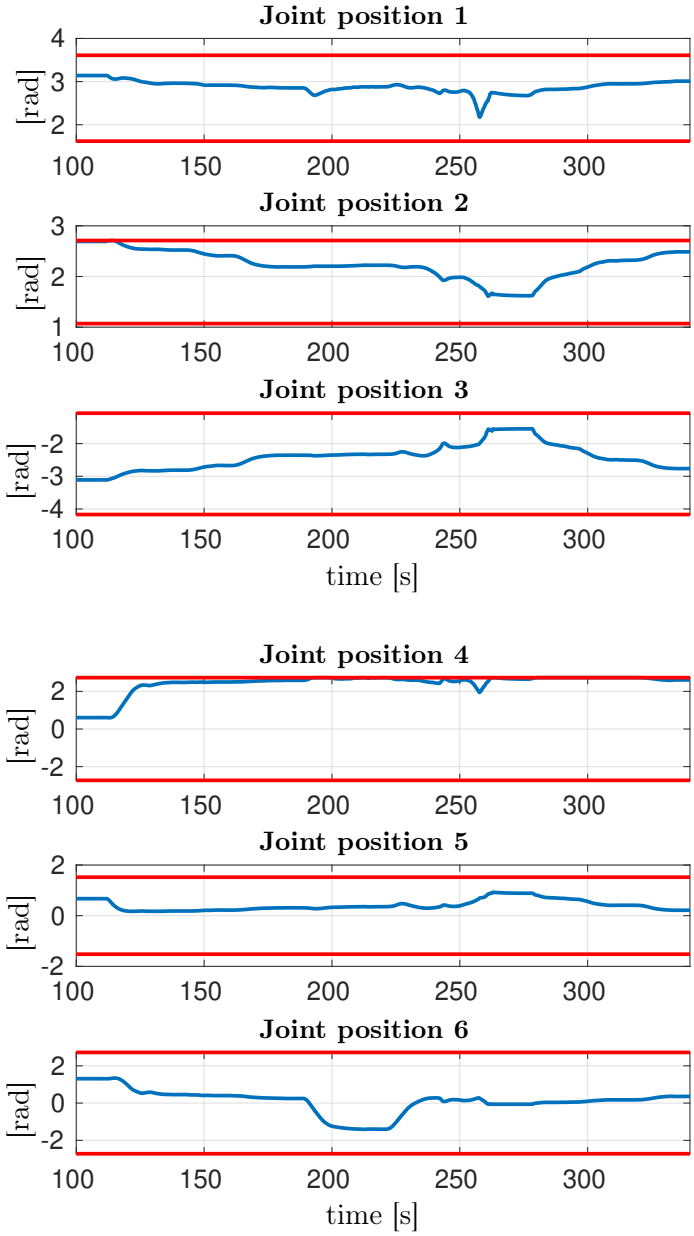

Figure 4.42: Turn valve with unexpected collisions: joint position over time (blue) and imposed thresholds (red).

Chapter 5

Motion Planner

5.1 Set-Based Task Constrained Motion Planning

In this thesis work a method based on merging a global planner and a local one is proposed. In detail, the global planner is structured into two steps. The first step is represented by the motion planner. It is a sampling-based algorithm which samples directly in the more intuitive and low dimension Cartesian space taking into account only the obstacle constraints. In this way the constraint representation is simpler and the search-space dimension is lower. The second step is represented by the simulation of the motion controller tracking the candidate trajectory from the motion planner. This action allows us to verify if the controller is able to track the generated reference trajectory fulfilling not only the task trajectory as in [56], [19] but also all the other equality/set-based task constraints and, in general, exploiting the redundancy. Indeed, the kinematic controller used in the global and local planner is the same. In this way, the real-time trajectory tracking by the local controller is guaranteed in case of static and perfectly known environment. Furthermore, during the movement the global planner is performed in background such that it can optimize the path or re-plan if a change in the environment occurs. The combination of these two planners allows us to take advantage from both ones. Indeed, the local minimum problem typical of local methods is overcome thanks to the global motion planner. On the other hand, the presence of a reactive local planner guarantees the task constraints are fulfilled while re-planning in real-time. In particular, as shown in the presented experiments, the proposed method allows

the system to run in real time with a realistic end-effector trajectory speed for a daily life scenario, with no need of specific circuitry.

5.1.1 Overall Control Algorithm Architecture

The control algorithm architecture is represented by the combination of a global and local planner, respectively, as shown in Fig. 5.1.

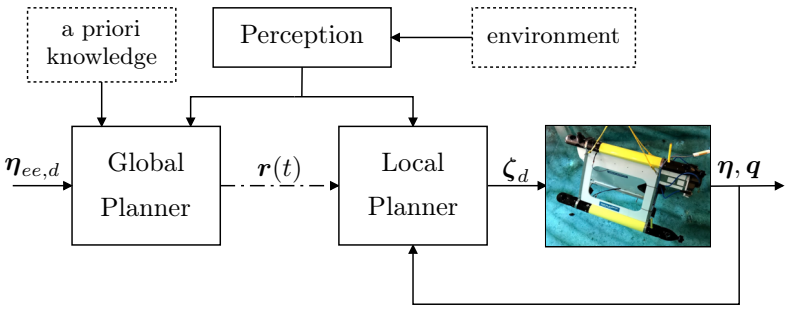

Figure 5.1: Overall control algorithm architecture. Given a desired 3D target $\boldsymbol{\eta}_{ee,d}$, the Global Planner outputs a new trajectory $\boldsymbol{r}(t)$ fulfilling all tasks constraints each time a path optimization or a mismatch between the a-priori knowledge and the information by the perception module is found (therefore it is not synchronous with the control architecture). Then the Local Planner computes the reference velocity $\boldsymbol{\zeta}_d$ for the system.

In particular, the global planner receives a 3D space desired target $\boldsymbol{\eta}_{ee,d}$ as input. Then, it has to find a feasible trajectory $\boldsymbol{r}(t)$ from the initial point to the desired one fulfilling both the Cartesian and joint-space task constraints. The local planner receives $\boldsymbol{r}(t)$ as input and computes the system reference velocity $\boldsymbol{\zeta}_d$ to make the system follow the desired trajectory while guaranteeing the constraints fulfillment.

Furthermore, during the system movement, the global planner runs in background taking into account the sensors feedback and continuously trying to optimize the path. Therefore, a new trajectory is sent to the local planner each time a path optimization is found or a re-planning is necessary because of mismatch between the a-priori information and the one acquired by the perception module, meaning that the global planner adapts to environment changes, e.g., obstacles and human presence.

It is worth noticing that the proposed architecture is modular and it can be easily applied to any robotic system.

5.1.2 Local planner

The local planner takes the output trajectory from the global one and it generates the reference system velocity taking into account several task constraints, e.g., obstacle avoidance, joint limits and virtual walls. In particular, it is represented by the Set-Based Task-Priority Inverse Kinematics (STPIK) algorithm, well described in Section 4.1. Therefore, taking into consideration a generic robotic system with n DOFs, there are several tasks that can be controlled in addition to its pose, e.g., the safety related tasks. Furthermore, several different task hierarchies can be composed according to the mission to perform.

5.1.3 Global planner

The global planner receives the 3D space desired pose $\boldsymbol{\eta}_{ee,d}$ as input and it computes a feasible trajectory $\boldsymbol{r}(t)$ satisfying all task constraints as output. More in detail, it is structured in two steps: the motion planner and the IK-check (inverse kinematics check) step, respectively (see Fig. 5.2).

The motion planner is a sampling-based algorithm which samples directly in the Cartesian space. Thus, the constraint representation is simplified and the computational load is much lower. In particular, the Rapidly-exploring Random Tree (RRT) Connect [43], based on building two trees from the start $\boldsymbol{\eta}_{ee,0}$ and goal point $\boldsymbol{\eta}_{ee,d}$, respectively, is used in this work. Nevertheless, other planning algorithms could be used. The motion planner performs taking into consideration the a-priori knowledge of the environment configuration and the information from the perception module. When a path $\boldsymbol{s}_{\text{path}}$, connecting $\boldsymbol{\eta}_{ee,0}$ to $\boldsymbol{\eta}_{ee,d}$ fulfilling the environment constraints and obstacles free, is found by the exploring algorithm, it is processed to obtain the trajectory $\boldsymbol{r}(t)$. In detail, a short-cutting iterative operation is performed on $\boldsymbol{s}_{\text{path}}$ between both consecutive and non-consecutive way-points. Thus, a check on the resulting path is performed to verify that the obstacle and environment constraints are

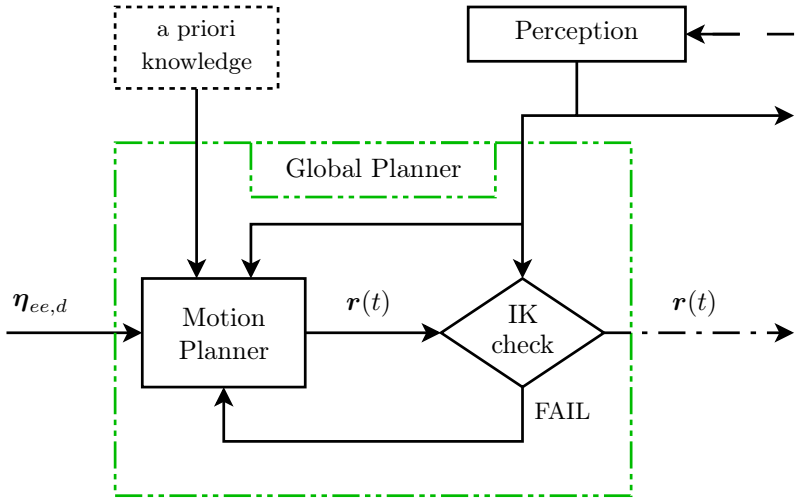

Figure 5.2: Global Planner structured in two steps: 1) the motion planner is a sampling-based algorithm which computes the trajectory $\boldsymbol{r}(t)$ to reach the desired target $\boldsymbol{\eta}_{ee,d}$ taking into consideration the a priori knowledge and the information from the perception module; 2) the IK-check, through the set-based task-prioirty inverse kinematics framework, verifies the trajectory tracking and the Cartesian/joint-space constraints fulfillment.

not violated. If they are not respected, the corresponding states are recovered. Finally a smoothing process is performed with B splines. Therefore, an interpolation operation is made on the resulting smoothed path with a specific desired sampling time T_s.

The output trajectory $\boldsymbol{r}(t)$ from the motion planner is sent to the IK-check which is the second step of the global planner. In this phase, the same control algorithm used for the local planner, that is the STPIK, is simulated taking $\boldsymbol{r}(t)$ as reference trajectory. It allows to verify not only the trajectory tracking but also the fulfillment of all the other Cartesian and joint space task constraints. Indeed, if some constraint is violated then the IK-check fails and the motion planner has to plan again. In this way, it is guaranteed that the constrained trajectory can be actually accomplished by the local controller in real-time. Nevertheless, if an environment change occurs, e.g., human presence, re-planning is necessary.

5.1.4 Re-plan details

The global planner, more specifically the motion planner, runs in background continuously trying to optimize the path and checking if the current trajectory $\boldsymbol{r}(t)$, together with the current information coming from the perception node, satisfies all the environment constraints.

When a violation is detected at, e.g., $t = t_A$, a re-plan is needed. Based on the algorithm and environmental parameters as well as a statistical survey, we known that a plan needs, in average, ΔT s. The planned robot state at $t_B = t_A + \Delta T$ is used by the motion planner in order to smoothly deviate from the current trajectory. It is worth noticing that, being the current trajectory verified by the IK-check, the state at t_B is consistent with the dynamic capability of the robot by construction.

The parameter ΔT may be also computed based on the position of the current violation, thus allowing a more conservative estimate and providing more computational time to the global planner.

There is not mathematical guarantee that the system is able to re-plan in time. This is rather obvious in case, e.g., an obstacle so far unknown to the perception node suddenly appears at a distance $\varepsilon << 1$ from any of the constraints. The comment above is rather obvious and physically consistent. However, by properly tuning the end-effector velocity with the bandwidth of the perception system and the CPU capability and by further assuming a maximum velocity to the moving obstacles this occurrence never happens. Nevertheless, if a proper tuning is not done, the local planner can guarantee the obstacle avoidance thanks to its reactive behaviour.

Algorithm 2 shows the pseudo-code of the proposed re-planning algorithm.

5.2 Numerical and Experimental Validations

Aimed at validating via numerical simulations ans real experiments the proposed approach, a ROS-based architecure has been developed. ROS [60], as already mentioned, is a popular middle-ware based on the publish-subscribe

Algorithm 2: Global_Planner($\boldsymbol{\eta}_{ee,d}$, perception, a-priori_knowledge)

```
while goal_not_reached do
    FAIL = true;
    while FAIL do
        [r(t), replan] = motion_planner(η_ee,d, perception);
        FAIL = IKcheck(r(t), perception);
    goal_not_reached = SystemState_check();
    if goal_not_reached then
        if replan then
            output(r(t));
        else
            optimized = path_optimiziation(r(t));
            if optimized then
                output(r(t));
    else
        continue;
```

paradigm that simplifies software testing, debug and integration for robotic applications.

For the perceptual feedback a Microsoft Kinect v2 that is a RGB-D sensor has been used. In particular, since the perception is not the focus of this work and being interested in the environment configuration, the distance from the objects has been computed through the ArUco library [62]. The latter is an Open Source library written in C++ for camera pose estimation using squared markers.

The robotic system used in the experimental setup is the Kinova Jaco2 7 DOFs with spherical wrist available in the Robotics Laboratory. It has been used as mockup in place of a UVMS since, from a kinematic point of view, the only significant difference is the fixed based. The latter makes the planing more complex because the arm is more subject to self-hits and kinematic singularities. The Kinova Jaco kinematic structure is represented in Fig. 5.3 and its Denavit-Hartenberg parameters are reported in Table 5.1. Being a 7 DOF robot, it is intrinsically redundant for any end-effector motion. It can be controlled via Universal Serial Bus (USB) or Ethernet link. In the specific case it is linked via Ethernet to a local network which allows to connect all the required

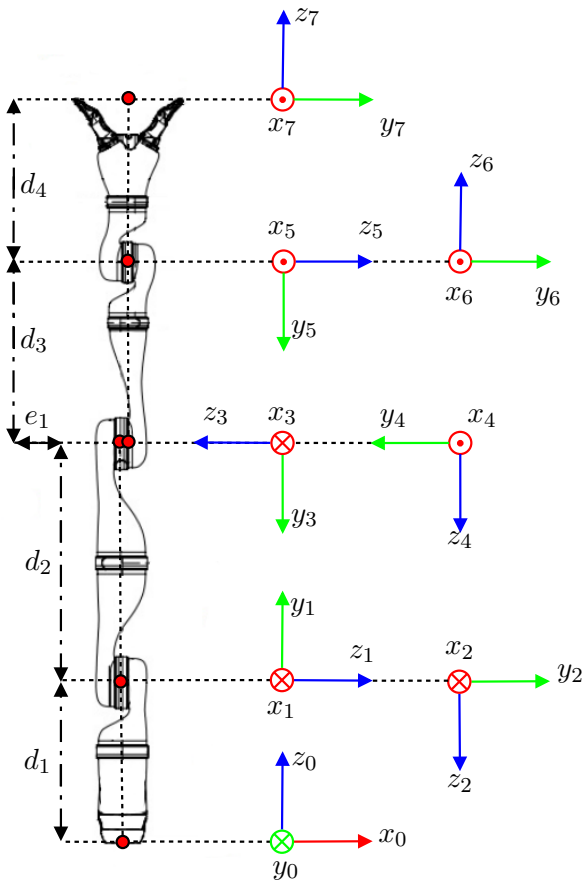

Figure 5.3: Kinova Jaco2 7 DOFs Spherical Wrist scheme with reference frames in Denavit-Hartenberg convention.

processes via both Ethernet and Wi-Fi. The presence of a local network shared by all the processes and devices allows to have a distributed system taking full advantage of the ROS framework and, thus, to run the perception and global/local planners on different hardware. In the specific case, the global and local planners are performed on a laptop with 4-core Intel i7-4510U processor clocked at 2.00 GHz and 8 GB of RAM.

The motion planner has been developed using the Open Motion Planning Library (OMPL) [72]. OMPL is a library containing many state-of-the-art sampling-based motion planning algorithms. It is also open source. Therefore, it has been possible to customize the existing functions to develop our own

joint	**a** [m]	$\boldsymbol{\alpha}$ [deg]	**d** [m]	$\boldsymbol{\theta}$ [deg]
1	0	$+90°$	0.2755	$+90°$
2	0	$+90°$	0	$0°$
3	0	$+90°$	-0.410	$0°$
4	0	$+90°$	-0.0098	$+180°$
5	0	$+90°$	-0.3111	$0°$
6	0	$+90°$	0	$0°$
7	0	$0°$	0.2638	$0°$

Table 5.1: Denavit-Hartenberg parameters of the Kinova Jaco2 7 DOFs Spherical Wrist Manipulator.

version of motion planner.

A library of control objectives has been developed in order to be available to the user that can compose them according to the specific mission:

- End-effector configuration control, equality-based task with required DOF typically $m = 3$ or $m = 6$;
- Distance of user-defined points of the robotic structure from any Cartesian point. Typically the user controls the distance of the end-effector the wrist and the elbow from obstacles acquired in the scene. Each task is a set-based task with $m = 1$;
- Cartesian virtual walls. Used to artificially confine the robot by restricting its movement. Each virtual wall is a set-based task with $m = 1$;
- Mechanical joint limits. Each task is a set-based task with $m = 1$. It is worth noticing that, for the Jaco2, rotation of q_1, q_3, q_5 and q_7 is not restricted;
- Robot self-hit. A set-based task that prevent the robot to touch its own structure. There are several possible implementation, for the Jaco2 we implemented a simple joint-space mapping. The task is a set-based one with $m = 1$;

- Optimization tasks such as manipulability, field of fiew or other direction sensor. They can be implemented as equality, set-based or gradient technique. Typically they are the lowest in priority.

In an intensive numerical simulation campaign all the tasks above have been combined in various ways in order to stress the *robustness* and generalization of the algorithm. Typically the priority follows the importance of the tasks, being the safety-tasks of higher priority, the middle are the mission-related tasks and the optimization the lowest.

Due to the selected ROS-based architecture and the properties of any kinematic controller, it is worth noticing that the sole difference from the numerical simulation and the experiment is given by the *virtual* of the perception node and the (limited) tracking error of the low level dynamic controller. All the code developed for the work at hand, thus, is invariant in the simulation and the experiment. The numerical simulation campaign is thus crucial to achieve some statistical information about the algorithm. Given the environment's dimension, equal to the robot workspace, an average distance of start to target point of 100 cm, the sampling time of 10 ms, the end-effector cruise velocity of 0.1 m/s, the number of DOFs $n = 7$ with a number of tasks varying from the sole end-effector configuration to the whole suite presented above, the global planning time was, in average, of 250 ms.

The number of re-planning strongly depends on the assumptions made on the perception system and the obstacle movement. It is somehow a condition that can be controlled and that we intentionally injected into the validation by, e.g., adding an obstacle at runtime unknown at the beginning of the movement. The re-planning time is affected by the same variable as the initial one but it is obviously lower as much as it is closer to the target.

In addition to numerical simulations, several real tests have been performed. The reported experiment consists of planning from an initial point to a desired one with obstacles configuration changes in the workspace during the movement (see Fig. 5.4). More in detail, the global planner initially plans taking into account an initial workspace configuration. Then, two of the four obstacles are moved in real-time along the manipulator path. Thus, it is necessary to re-plan

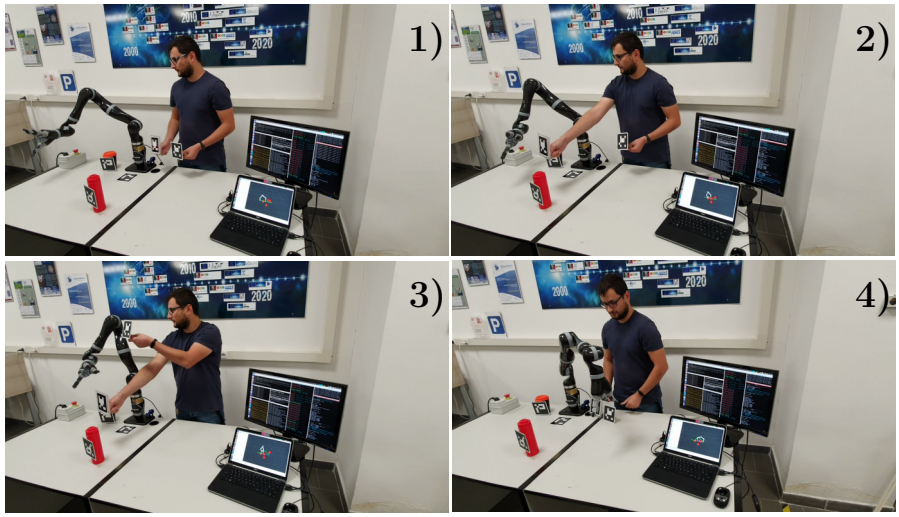

Figure 5.4: Frames of the experiment.

to find new feasible paths that are collisions free. In the meantime, the local planner, which is a reactive behaviour algorithm, ensures that all Cartesian and joint-space constraints are fulfilled and, therefore, no collision happens. When the re-planning is successful, the global planner continues to run trying to optimize the path.

Figure 5.5 and Fig. 5.6.a show the end-effector, wrist and elbow, respectively, distance values from the four obstacles present in the scene. Figure 5.6.b reports the joint limits and Fig. 5.7 shows the end-effector and wrist distances from the second and third joint, namely the self-hits. In particular, despite the changes in the workspace during the system movement, it is noticeable how the several constraints in the joint and Cartesian space are fulfilled thanks to the set-based task-priority inverse kinematics control algorithm which guarantees the feasibility of the trajectory from the global planner without the risk to get trapped in local minimum.

Graphical renderings of some numerical simulations along with the video of the proposed experiment are available at www.danieledivitoengineering.it.

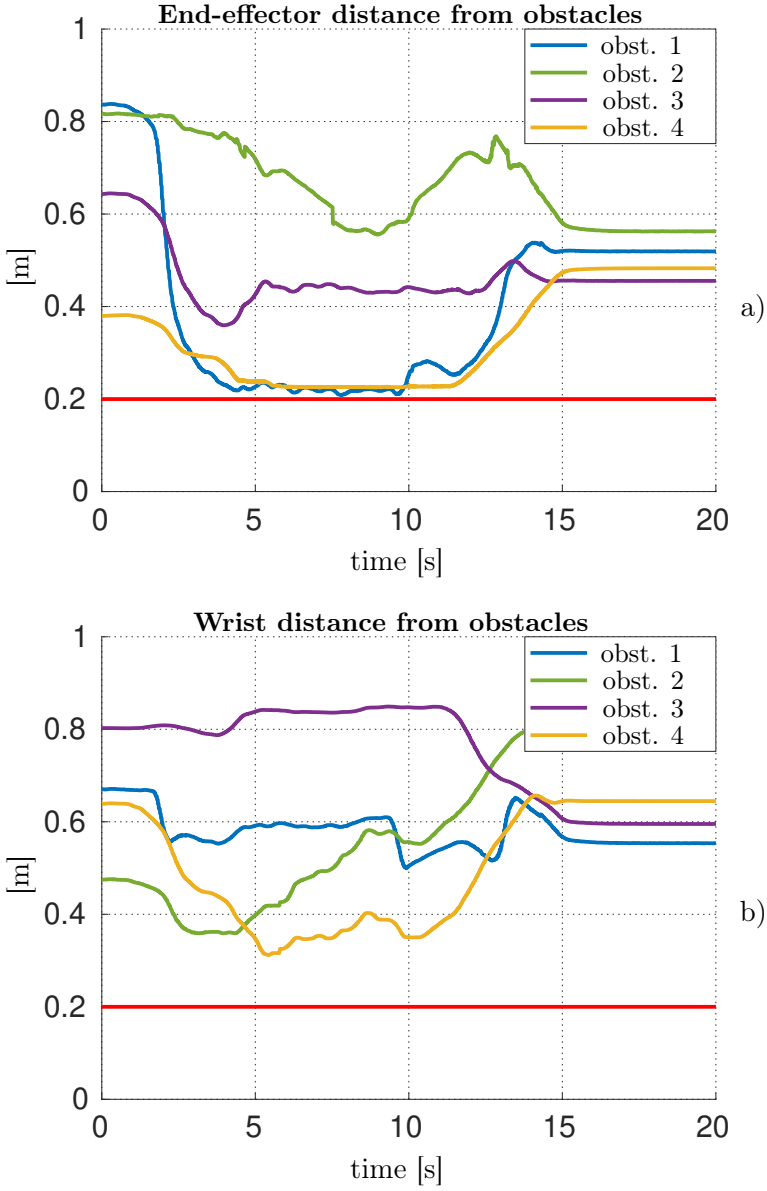

Figure 5.5: Experiment results - safety related (set-based) tasks: a) end-effector distance from obstacles; b) wrist distances from obstacles.

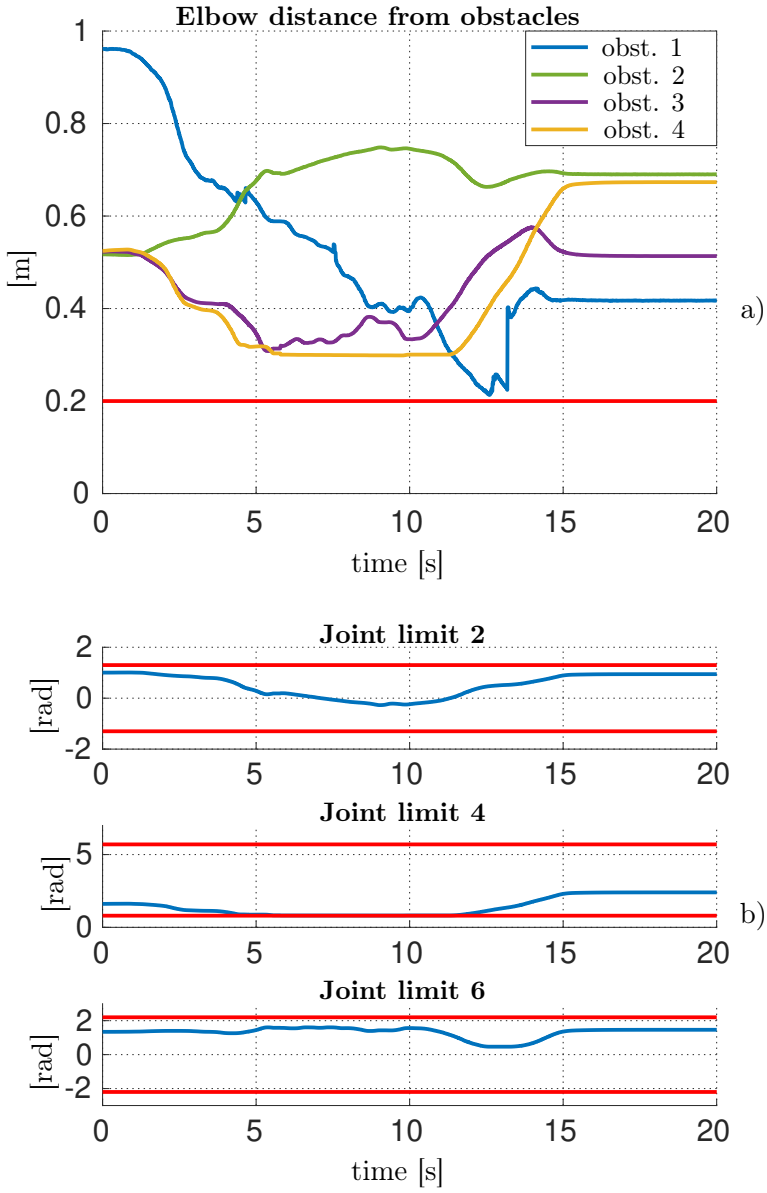

Figure 5.6: Experiment results - safety related (set-based) tasks: a) elbow distance from obstacles; b) joints limits.

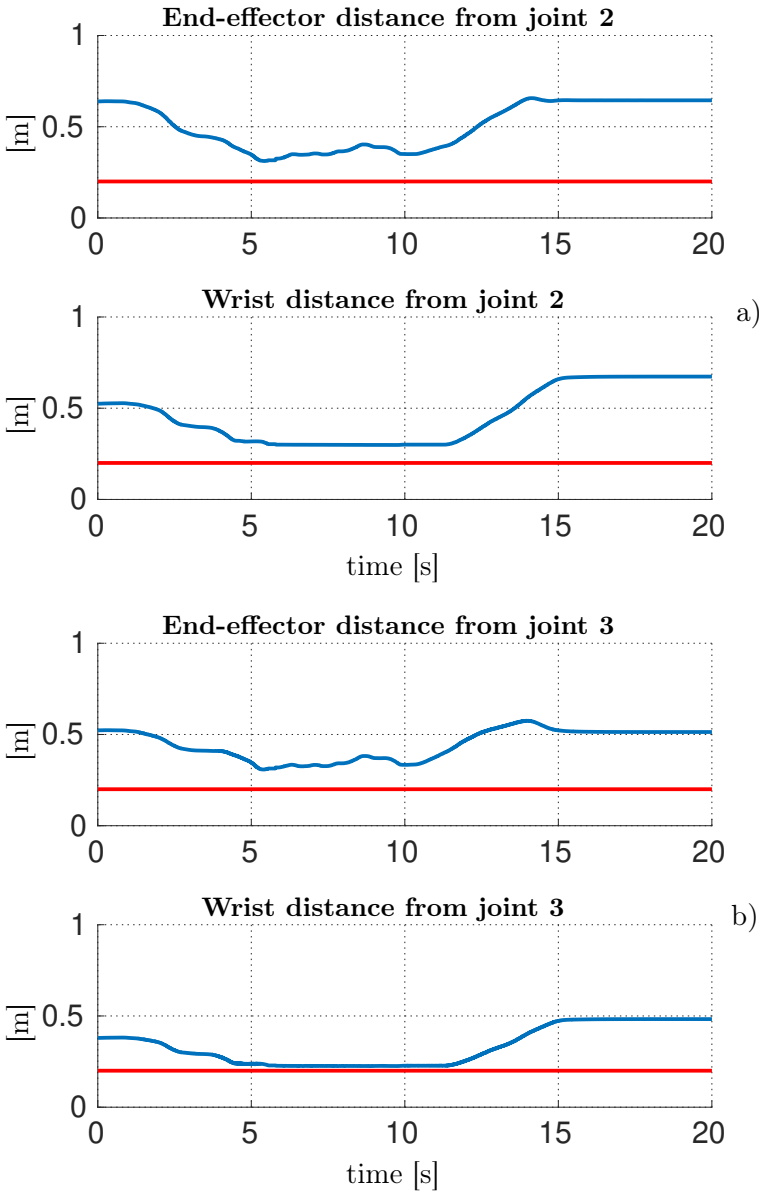

Figure 5.7: Experiment results - safety related (set-based) tasks: a) end-effector and wrist distances from the 2nd joint; b) end-effector and wrist distances from the 3rd joint.

5.2.1 Conclusions

A method based on merging a global and local planner for redundant robots has been proposed. The key idea is to combine the properties of both methods compensating their limits: the computational payload for the global planner and the local minimum problem for the local planner. The proposed approach has been validated experimentally with a Kinova Jaco2 7 DOFs.

The approach is currently being extended to structures with higher DOFs, such as the Kinova Movos available in the lab and floating-base structures such as aerial and underwater. Furthermore, a formal proof, guaranteeing convergence under a stochastic framework will be addressed next.

It is worth remarking that the current approach uses all the DOFs of the redundant robot in a task-priority approach. As possible future activity it is worth investigating the use of the null-space to explore self-motion in case a trajectory is unfeasible similar to what done in [56], [19].

Chapter 6

Conclusions and future work

This thesis work has focused on the whole control architecture of a UVMS, taking into analysis the following control layers:

- *Dynamic Control* - it is the lowest control level which is has to compute the forces/torques necessary to compensate the hydrodynamic effects and external disturbances in order to make the system follow the reference velocity.
- *Kinematic Control* - it is the middle level control that computes the reference velocities that make the system follow the desired trajectory.
- *Motion Planner* - it is the highest control level is in charge of planning the motion given a desired pose taking into consideration the system and environment constraints.

Regarding the dynamic layer, aimed at compensating the persistent dynamic effect, e.g., the system gravity/buoyancy and the ocean current, in the proper reference frame, an adaptive control algorithm has been proposed. Furthermore, considering a UVMS, the manipulator movements cause disturbance on the vehicle. Then, a recursive adaptive algorithm able to counteract for the manipulator interactions has been presented. A deep investigation of the thruster dynamics effects on the control loop has been conducted as well, showing the *distortion* effects that can have on the identified parameters and the control gains. The adaptive control has been validated through numerical simulations taking into account AUV and UVMS. Moreover, it has been experimentally

validated within the ROBUST project doing a comparison with a PI controller which resulted in a better performance of the adaptive control.

A future work would be to split the dynamic parameters, identified during the learning phase, from the actual integral action such that to not make the parameters loose their physical meaning during the transient time as in the way points navigation with no imposed trajectory.

Within the aim to take into account safety related tasks, the Set-based Task-Priority Inverse Kinematics (STPIK) control algorithm has been presented. Indeed, this algorithm allows to manage directly in the kinematic layer not only the operation tasks but also all the safety-related tasks fulfilling the joint and Cartesian space constraints. Furthermore, a detailed analysis on DLS algorithms for managing kinematic singularities has been conducted as well. The effectiveness of the STPIK algorithm has been proved through its application in the simulation of an assistive control framework for ROV piloting and within the DexROV project. In the latter case, the inverse kinematics algorithm has been also combined with an admittance control resulting in a safe control architecture for the semi-autonomous vehicle-manipulator system.

The proposed STPIK algorithm has a major drawback represented by the sudden activation/deactivation of the set-based tasks. Indeed, the latter induces a discontinuity of the joint velocities in output of the algorithm, that may ask for high joint acceleration. Thus, a future work would be to implement a method for smoothly managing the task activation/deactivation transitions while maintaining the strict priority among the tasks.

The inverse kinematics controller is a local method and, therefore, it is subject to local minimum. One possible solution is represented by the use of a global planner. However, the latter requires a high computational load and, thus, it is usually not able to perform fast re-plan in real-time. Thus, aimed at performing trajectories with no local minimum and at guaranteeing the system and workspace safety, a method based on merging the global planner with the local one, that is the STPIK, has been proposed. The effectiveness of this approach has been experimentally proved using a 7 DOFs Kinova manipulator as mockup.

Future work about the proposed Motion planner approach will consist in its application to mobile robots with higher DOFs and, then, taking into consideration a larger workspace. Furthermore, a deep comparison with other state of the art planners will be performed and a formal proof, guaranteeing convergence under a stochastic framework will be addressed as well.

Bibliography

[1] SNAME, Nomenclature for treating the motion of a submerged body through a fluid. *Technical and Research Bulletin*, pages 1–5, 1950.

[2] DexROV website. http://www.dexrov.eu/, 2015. [Online; accessed 20-January-2020].

[3] ROBUST website. http://eu-robust.eu, 2016. [Online; accessed 20-January-2020].

[4] The efficiencies of low logistics, man-portable auvs for shallow water survey operations. http://www.subseauk.com/documents/ncs%20survey.pdf, 2017.

[5] B.D. Anderson. Failures of adaptive control theory and their resolution. *Communications in Information & Systems*, 5(1):1–20, 2005.

[6] G. Antonelli. On the use of adaptive/integral actions for six-degrees-of-freedom control of autonomous underwater vehicles. *IEEE Journal of Oceanic Engineering*, 32(2):300–312, April 2007.

[7] G. Antonelli and E. Cataldi. Recursive adaptive control for an underwater vehicle carrying a manipulator. In *22nd Mediterranean Conference on Control and Automation*, pages 847–852, June 2014.

[8] G. Antonelli, S. Chiaverini, N. Sarkar, and M. West. Adaptive control of an autonomous underwater vehicle: experimental results on odin. *IEEE Transactions on Control Systems Technology*, 9(5):756–765, Sep. 2001.

[9] Gianluca Antonelli and G Antonelli. *Underwater robots*, volume 4. Springer, 2018.

[10] Filippo Arrichiello, Paolo Di Lillo, Daniele Di Vito, Gianluca Antonelli, and Stefano Chiaverini. Assistive robot operated via p300-based brain computer interface. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6032–6037. IEEE, 2017.

[11] P. Baerlocher. *Inverse kinematics techniques for the interactive posture control of articulated figures*. PhD thesis, École Polytechnique Fédéral De Lausanne, 2001.

[12] Dmitry Berenson, Siddhartha Srinivasa, and James Kuffner. Task space regions: A framework for pose-constrained manipulation planning. *The International Journal of Robotics Research*, 30(12):1435–1460, 2011.

[13] Dmitry Berenson, Siddhartha S Srinivasa, Dave Ferguson, and James J Kuffner. Manipulation planning on constraint manifolds. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 625–632, 2009.

[14] G. Buizza Avanzini, A. Zanchettin, and P. Rocco. Constrained model predictive control for mobile robotic manipulators. *Robotica*, 36(1):19–38, 2018.

[15] Heiko Bülow and Andreas Birk. Spectral 6dof registration of noisy 3d range data with partial overlap. *IEEE transactions on pattern analysis and machine intelligence*, 35(4):954–969, 2013.

[16] F. Caccavale, S. Chiaverini, and B. Siciliano. Second-order kinematic control of robot manipulators with jacobian damped least-squares inverse: theory and experiments. *IEEE/ASME Transactions on Mechatronics*, 2(3):188–194, Sep 1997.

[17] M. Caccia, G. Indiveri, and G. Veruggio. Modeling and identification of open-frame variable configuration unmanned underwater vehicles. *Oceanic Engineering, IEEE Journal of*, 25(2):227–240, 2000.

[18] Sylvain Calinon, Danilo Bruno, and Darwin G Caldwell. A task-parameterized probabilistic model with minimal intervention control. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3339–3344. IEEE, 2014.

[19] M. Cefalo and G. Oriolo. A general framework for task-constrained motion planning with moving obstacles. *Robotica*, 37(3):575–598, 2019.

[20] M. Cefalo, G. Oriolo, and M. Vendittelli. Planning safe cyclic motions under repetitive task constraints. In *2013 IEEE International Conference on Robotics and Automation*, pages 3807–3812, May 2013.

[21] M. Cefalo, G. Oriolo, and M. Vendittelli. Task-constrained motion planning with moving obstacles. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5758–5763, Nov 2013.

[22] N. M. Ceriani, A. M. Zanchettin, P. Rocco, A. Stolt, and A. Robertsson. Reactive task adaptation based on hierarchical constraints classification for safe industrial robots. *IEEE/ASME Transactions on Mechatronics*, 20(6):2935–2949, 2015.

[23] S. Chiaverini. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Transactions on Robotics and Automation*, 13(3):398–410, June 1997.

[24] S. Chiaverini, G. Oriolo, and I. D. Walker. *Springer Handbook of Robotics*, chapter Kinematically Redundant Manipulators, pages 245–268. B. Siciliano, O. Khatib, (Eds.), Springer-Verlag, Heidelberg, D, 2008.

[25] S. Chiaverini, B. Siciliano, and O. Egeland. Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator. *IEEE Transactions on Control Systems Technology*, 2(2):123–134, June 1994.

[26] Nancy J Cooke. Human factors of remotely operated vehicles. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 50, pages 166–169, 2006.

[27] M. L. Corradini and G. Orlando. A discrete adaptive variable-structure controller for mimo systems, and its application to an underwater rov. *IEEE Transactions on Control Systems Technology*, 5(3):349–359, May 1997.

[28] Arati S. Deo and Ian D. Walker. Overview of damped least-squares methods for inverse kinematics of robot manipulators. *Journal of Intelligent and Robotic Systems*, 14(1):43–68, 1995.

[29] P. Di Lillo, E. Simetti, D. De Palma, E. Cataldi, G. Indiveri, G. Antonelli, and G. Casalino. Advanced ROV autonomy for efficient remote control in the DexROV project. *Marine Technology Society Journal*, 50(4):67–80, 2016.

[30] Paolo Di Lillo, Filippo Arrichiello, Gianluca Antonelli, and Stefano Chiaverini. Safety-related tasks within the set-based task-priority inverse kinematics framework. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6130–6135. IEEE, 2018.

[31] Gregory Dudek and Michael Jenkin. *Computational principles of mobile robotics*. Cambridge university press, 2010.

[32] J. C. Evans, K. M. Keller, J. S. Smith, P. Marty, and O. V. Rigaud. Docking techniques and evaluation trials of the swimmer auv: an autonomous deployment auv for work-class rovs. In *Proceedings of MTS/IEEE Oceans 2001*, volume 1, pages 520–528, 2001.

[33] T. I. Fossen and J. G. Balchen. The nerov autonomous underwater vehicle. In *OCEANS 91 Proceedings*, pages 1414–1420, Oct 1991.

[34] Thor I. Fossen. *Marine Control Systems: Guidance, Navigation, and Control of Ships, Rigs and Underwater Vehicles*. 2002.

[35] T.I. Fossen. *Marine Control Systems: Guidance, Navigation and Control of Ships, Rigs and Underwater Vehicles*. Marine Cybernetics, Trondheim, Norway, 2002.

[36] M. Gautier and W. Khalil. Direct calculation of minimum set of inertial parameters of serial robots. *IEEE Transactions on Robotics and Automation*, 6:368–373, 1990.

[37] John-Morten Godhavn, Thor I Fossen, and Svein P Berge. Non-linear and adaptive backstepping designs for tracking control of ships. *International Journal of Adaptive Control and Signal Processing*, 12(8):649–670, 1998.

[38] Ioannis Havoutis and Sylvain Calinon. Learning from demonstration for semi-autonomous teleoperation. *Autonomous Robots*, 43(3):713–726, Mar 2019.

[39] Anthony J Healey and David Lienard. Multivariable sliding mode control for autonomous diving and steering of unmanned underwater vehicles. *IEEE journal of Oceanic Engineering*, 18(3):327–339, 1993.

[40] G. Indiveri and G. Parlangeli. On thruster allocation, fault detection and accomodation issues for underwater robotic vehicles. In *Proc. Int. Symp. Communications, Control and Signal Processing, Marrakech*, 2006.

[41] Oussama Kanoun, Florent Lamiraux, and Pierre-Brice Wieber. Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task. *IEEE Transactions on Robotics*, 27(4):785–792, 2011.

[42] Hassan K Khalil. Nonlinear systems. *Upper Saddle River*, 2002.

[43] J. J. Kuffner and S. M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 2, pages 995–1001 vol.2, April 2000.

[44] S. La Valle. *Planning Algorithms*. Cambridge University Press, 2006.

[45] Martin Ludvigsen, Geir Johnsen, Asgeir J Sørensen, Petter A Lågstad, and Øyvind Ødegård. Scientific operations combining rov and auv in the trondheim fjord. *Marine Technology Society Journal*, 48(2):59–71, 2014.

[46] A.A. Maciejewski. Numerical filtering for the operation of robotic manipulators through kinematically singular configurations. *Journal of Robotic Systems*, 5(6):527–552, 1988.

[47] Anthony A Maciejewski and Charles A Klein. Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *The international journal of robotics research*, 4(3):109–117, 1985.

[48] N. Mansard, O. Khatib, and A. Kheddar. A unified approach to integrate unilateral constraints in the stack of tasks. *IEEE Transactions on Robotics*, 25(3):670–685, June 2009.

[49] G. Marani, Jinhyun Kim, Junku Yuh, and Wan Kyun Chung. A real-time approach for singularity avoidance in resolved motion rate control of robotic manipulators. In *Proc. IEEE International Conference on Robotics and Automation ICRA '02*, volume 2, pages 1973–1978, 2002.

[50] Giacomo Marani, Song K Choi, and Junku Yuh. Real-time center of buoyancy identification for optimal hovering in autonomous underwater intervention. *Intelligent Service Robotics*, 3(3):175–182, 2010.

[51] Signe Moe, Gianluca Antonelli, Andrew R Teel, Kristin Y Pettersen, and Johannes Schrimpf. Set-based tasks within the singularity-robust multiple task-priority inverse kinematics framework: General formulation, stability analysis, and experimental results. *Frontiers in Robotics and AI*, 3:16, 2016.

[52] S. Murray, W. Floyd-Jones, , Y. Qi, D.J. Sorin, and G. Konidaris. Robot motion planning on a chip. In *Robotics: Science and Systems*, 2016.

[53] Giuseppe Muscio, Francesco Pierri, Miguel Angel Trujillo, Elisabetta Cataldi, G Giglio, Gianluca Antonelli, Fabrizio Caccavale, Antidio Viguria, Stefano Chiaverini, and Aníbal Ollero. Experiments on coordinated motion of aerial robotic manipulators. In *Proceedings IEEE International Conference onRobotics and Automation*, pages 1224–1229, 2016.

[54] Yoshihiko Nakamura and Hideo Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of dynamic systems, measurement, and control*, 108(3):163–171, 1986.

[55] Yoshihiko Nakamura, Hideo Hanafusa, and Tsuneo Yoshikawa. Task-priority based redundancy control of robot manipulators. *The International Journal of Robotics Research*, 6(2):3–15, 1987.

[56] G. Oriolo, M. Cefalo, and M. Vendittelli. Repeatable motion planning for redundant robots over cyclic tasks. *IEEE Transactions on Robotics*, 33(5):1170–1183, Oct 2017.

[57] G. Oriolo and M. Vendittelli. A control-based approach to task-constrained motion planning. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 297–302, Oct 2009.

[58] Alistair Palmer, Grant E. Hearn, and Peter Stevenson. Modelling tunnel thrusters for autonomous underwater vehicles. *IFAC Proceedings Volumes*, 41(1):91 – 96, 2008.

[59] Max Pfingsthorn, Andreas Birk, and Heiko Buelow. Uncertainty estimation for a 6-dof spectral registration method as basis for sonar-based underwater 3d slam. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3049–3054. IEEE, 2012.

[60] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[61] JS Riedel. Shallow water stationkeeping of an autonomous underwater vehicle: the experimental results of a disturbance compensation controller. In *OCEANS 2000 MTS/IEEE Conference and Exhibition. Conference Proceedings (Cat. No. 00CH37158)*, volume 2, pages 1017–1028. IEEE, 2000.

[62] Francisco J. Romero-Ramirez, Rafael Muñoz-Salinas, and Rafael Medina-Carnicer. Speeded up detection of squared fiducial markers. *Image and Vision Computing*, 76:38 – 47, 2018.

[63] A. Saunders and M. Nahon. The effect of forward vehicle velocity on through-body auv tunnel thruster performance. In *OCEANS '02 MTS/IEEE*, volume 1, pages 250–259 vol.1, Oct 2002.

[64] A. Shkolnik and R. Tedrake. High-dimensional underactuated motion planning via task space control. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3762–3768, Sept 2008.

[65] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2016.

[66] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[67] Enrico Simetti and Giuseppe Casalino. A novel practical technique to integrate inequality control objectives and task transitions in priority based control. *Journal of Intelligent & Robotic Systems*, 84(1-4):877–902, 2016.

[68] Enrico Simetti, Giuseppe Casalino, Sandro Torelli, Alessandro Sperinde, and Alessio Turetta. Floating underwater manipulation: Developed control methodology and experimental validation within the trident project. *Journal of Field Robotics*, 31(3):364–385, 2014.

[69] Siciliano B Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In *proceeding of 5th International Conference on Advanced Robotics*, volume 2, pages 1211–1216, 1991.

[70] Ø. N. Smogeli. *Control of Marine Propellers - From Normal to Extreme Conditions*. PhD thesis, Department of Marine Technology, NTNU, Trondheim, Norway, 2006.

[71] M. Stilman. Global manipulation planning in robot joint space with task constraints. *IEEE Transactions on Robotics*, 26(3):576–584, June 2010.

[72] Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012. http://ompl.kavrakilab.org.

[73] T. Sugihara. Solvability-unconcerned inverse kinematics by the levenberg marquardt method. *IEEE Transactions on Robotics*, 27(5):984–991, Oct 2011.

[74] M. Tognon, E. Cataldi, H.A. Tello Chávez, G. Antonelli, J. Cortes, and A. Franchi. Control-aware motion planning for task-constrained aerial manipulation. *IEEE Robotics and Automation Letters*, 3(3):2478–2484, 2018.

[75] D. R. Yoerger, J. G. Cooke, and J. J. E. Slotine. The influence of thruster dynamics on underwater vehicle behavior and their incorporation into control system design. *IEEE Journal of Oceanic Engineering*, 15(3):167–178, Jul 1990.

[76] T. Yoshikawa. Manipulability and redundancy control of robotic mechanisms. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1004–1009. IEEE, 1985.

[77] J. Yuh. Learning control for underwater robotic vehicles. *IEEE Control Systems Magazine*, 14(2):39–46, April 1994.

[78] Tomasz Łuczyński, Tobias Doernbach, Shashank Govindaraj, Christian Mueller, and Andreas Birk. 3d grid map transmission for underwater mapping and visualization under bandwidth constraints. In *OCEANS 2017-Anchorage*, 09 2017.