



UNIVERSITÀ DEGLI STUDI
DI GENOVA

Department of Informatics, Bioengineering, Robotics and Systems
Engineering

Cooperative Robotic Manipulation for the Smart Factory

Francesco Wanderlingh

Advisor: Prof. Giuseppe Casalino

Co-advisor: Prof. Enrico Simetti

A thesis submitted in partial fulfilment of the
PhD Program in Robotics and Autonomous Systems

Apr, 3rd 2018

2018 © Francesco Wanderlingh
GRAAL (Genoa Robotics and Automation Laboratory)
DIBRIS DEPARTMENT, UNIVERSITY OF GENOVA
Via All'Opera Pia, 13
Genova (GE), 16145
Italia

WWW.GRAAL.DIBRIS.UNIGE.IT

For any information about this publication you can contact me, the author, at *fwanderlingh@gmail.com*. This document uses the Charter font family and has been proudly typeset in $\text{\LaTeX}$.

First printing, June 2018

*To my family,
the one of blood
and the one of luck*

Acknowledgements

First of all I would like to thank my advisor, Pino (a.k.a. Prof. Casalino), for the curiosity and energy he passed on me, the knowledge and expertise he, always, willingly, shared, and for the trust he gave me in the first place.

Secondly, this thesis would not have been possible without the additional guidance of my co-advisor Enrico Simetti, which has made his support available in an uncountable number of ways. Assisting me, despite my *vibrancy* ;), through sessions of debugging and various conundrum solving discussions. With his eagle eye he, almost, always got me out of trouble, getting all the *bits* in line.

It is also a pleasure to thank my lab mates Alessandro Sperindé, Sandro Torelli and Ninad Manerikar, for sharing with me this journey, the experimental trials, dispensing good advices about programming, but also about tinkering and music. And, let's say it, for making me a much more better nerd.

A special thanks goes to my sweetheart, Fabiana, and my family, who have always believed in me and gave me support, through all the struggles that one can face in growing up while doing a PhD ;).

Finally I would like to thank for the resources offered by the GRAAL and EmaroLab laboratories, that made possible for my ideas and code to be effectively tested on real hardware, which is one of the most rewarding moments for every engineer.



Abstract

To tackle the upcoming needs of the new Industry 4.0 paradigm, where cyber-physical systems communicate and cooperate with each other and humans in real time, the control of a robot must be able to guarantee a high level of flexibility, in order to accomplish a variety of tasks that go beyond the usual, standardized, robotic cell ones taking place within a very structured environment. Stemming from the well-established line of research on control of high degrees of freedom marine systems investigated in the GRAAL Lab where I carried out my PhD, the focus of this work has been to develop a general robot control framework to transfer the know-how acquired in years of marine research to industrial scenarios.

In the perspective of building a solution that *fits them all*, removing the need to create specific solutions for every single problem, the main idea has been to develop a framework that provides a set of basic control *capabilities*, as building blocks for the robots to perform complex actions. In this context the PhD work focused on the development of a *real time control framework* and adaptation of the involved structures, to develop an experimentally working set up that is capable of: handling different robotic structures (fixed base arms, single and dual arm mobile manipulators either working on ground or underwater) in a uniform way; handling cooperation between different agents; being able to perform interaction tasks with the environment and human beings; exposing interfaces to guarantee a good integration with the higher levels of planning, decision and command, with the goal of releasing them from dealing with lower level control problems. From the technological point of view, a key objective has been to develop the software architecture, implementing the control framework, in order to achieve the highest possible re-usability and flexibility. This allowed an easy application of the control framework to different robotic setups, minimizing the development effort to adapt not only the control laws, but also all the collateral supporting software needed to interact with the specific robot-software environment. To the above aims I exploited different hardware resources such as two YouBots platforms and a Baxter bi-manual robotic system, collecting experimental results which validate the proposed approach.

TABLE OF CONTENTS

Nomenclature	viii
1 Introduction	1
1.1 State of the art and innovation	3
1.2 Scientific and Technological Objectives	6
1.3 Experimental Outline	11
2 Single Agent Control: Theory	14
2.1 General Definitions	16
2.1.1 Notation	16
2.2 Definitions	16
2.3 Control Objectives	17
2.3.1 Reactive Control Tasks	18
2.3.2 Non-reactive Control Tasks	19
2.4 Actions and Missions	19
2.5 Activation Functions	20
2.5.1 Action Transition	22
2.6 Task Priority Inverse Kinematics	23
2.6.1 Remarks on Conflicting Objectives	25
2.7 Control in Presence of Vehicle Underactuations	26
2.8 Vehicle-Arm Coordination Scheme	28
2.9 Vehicle-Arm Multi-rate Control	30
3 Single Agent Control: Applications	31
3.1 Single-Arm Control	32
3.1.1 Experimental Results	35
3.2 Dual-Arm Control	36

3.2.1	Experimental Results	39
3.3	Mobile Manipulator Control	45
3.3.1	Simulation Results	48
4	Extension to Multi-Agent: Theory	52
4.1	Coordination Algorithm	54
4.1.1	First Independent Optimization	55
4.1.2	Coordination phase	56
4.1.3	Second Independent Optimization	58
4.1.4	Condition for Priority Inversion Invariance	58
5	Multi-Agent Control: Applications	60
5.1	Floating Base Cooperative Transportation	60
5.1.1	Simulation Results	63
5.2	Ground Base Cooperative Transportation	69
5.2.1	Experimental Results	70
	Conclusions	73
	Appendix The Software Architecture	77
A.1	Controller Modules	78
A.1.1	External Libraries	82
A.2	Graphical User Interface	83
	List of Figures	86
	References	89

NOMENCLATURE

API	Application Programming Interface
DOF	Degrees of Freedom
DCL	Dynamic Control Layer
GUI	Graphical User Interface
HCI	Human Computer Interface
HLL	High Level Layer
iCAT	inequality Control objectives, Activations and Transitions
JCB	Joint Control Boards
KCL	Kinematic Control Layer
ROS	Robot Operating System
ROV	Remotely Operated Vehicle
RTOS	Real Time Operating System
TCP	Tool Centre Point
TPIK	Task Priority Inverse Kinematics
UDP	User Data Protocol
UVMS	Unmanned Vehicle-Manipulator System

Control tasks category identifiers

r/nr	Reactive/Non-reactive task
=/≤	Equality/Inequality task
C	Physical Constraint task
S	Safety task
P	Prerequisite task
A	Action-defining task
O	Optimization task

INTRODUCTION

ROBOTS have been massively introduced in the field of manufacturing around the eighties, but ever since they have been mainly used in very fixed and structured environments and with no or rare interaction of robots with human workers. Nowadays new kind of robotic manipulators, including dual arm setups and mobile manipulators, are being rolled out from bigger producers such as KUKA, ABB, Universal Robots, Rethink Robotics and more, featuring lightweight and compliant structures that can be exploited to tackle the upcoming needs of the new Industry 4.0 paradigm where cyber-physical systems communicate and cooperate with each other and humans in real time. These trends have been especially identified, in the European Multi-Annual Roadmap for robotics, as innovations with high potential impact on market domains [1] and covered in a series EU H2020 calls, such as the recent ICT (Information and Communication Technology) one on *Advanced robot capabilities* research [2]. Towards that end, the control of the robot must be able to guarantee a high level of flexibility in order to accomplish a variety of tasks that go beyond the usual, standardized, robotic cell ones, which use to take place within a very structured environment. The basic idea is that the robot should autonomously take care of all the control objectives related to the safety of the system, interaction with the environment, as well as those that are a prerequisite to accomplish the required task. Achieving these objectives would indeed grant the robot a higher level of flexibility, simplifying the higher levels that deal with planning and task sequencing.

This PhD work is in fact framed within the Italian technology cluster *Fabbrica Intelligente* (i.e. Smart Manufacturing). In particular, as seen in figure 1.1, in the

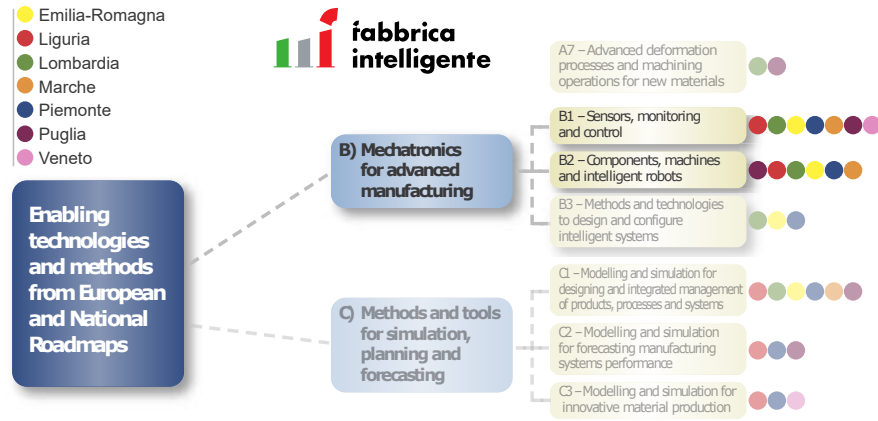


Figure 1.1: A focused extract from the Italian “*Research and Innovation Roadmap*” [3] for smart manufacturing, highlighting the scope in which this PhD work is framed.

Italian “*Research and Innovation Roadmap*” [3], a series of the key enabling technologies have been identified in the area of “*Mechatronics for advanced manufacturing*”, where the Liguria region in which our university resides, is involved in the development of *Sensors, monitoring and control* and *Components, machines and intelligent robots*.

A valid perspective, toward non-negligible improvements of the operational activities in such fields, is the employment of cooperating robots. The ‘cooperating’ term is a relatively new introduction in the field of robotics, and it has mainly two meanings: being *collaborative* with an operator in human-robot interaction (HRI) scenarios, or, to *coordinate* with a set of other robotic structures. Research in such two fields aims at improving the execution complex activities within factory and logistic scenarios. In this work we will focus more on the coordination aspect of the problem. In particular, employing cooperating robots may imply the coordination between two arms within a dual-arm work-cell, or the coordination of a mobile platform with supported arms, within vehicle-manipulator systems used for de-localized interventions; up to the coordination between different vehicle-manipulator systems employed for transporting large-space and/or large-weight items. Basic robotic structures, or basic units (i.e. vehicles and arms), could be employed either as individual agents or as coordinately operating parts within more complex organizations of them, characterized by enhanced overall operational capabilities. In this perspective this should also lead to higher operational versatility such as expandability and scalability via reconfigurations. In section 1.2 we will analyse this aspect more in detail.

What will consider as our final scenario, is a somewhat generic factory or logistic environment, employing different mechanically organized robotic structures, ranging from the simplest ones, like vehicles and fixed-base single-manipulators, till the most complex ones, as for instance vehicle/dual-arm systems, will be assumed; where clusters of them can also be employed for executing coordinated activities beyond those allowed to individual robotic structures.

1.1 State of the art and innovation

Redundant robotic systems are becoming increasingly more common than a few decades ago. As these systems increased with popularity, the computational power available to control them has scaled. The increased computational power has allowed to exploit the high versatility offered by such systems, due to their high number of degrees of freedom, in order to perform different, complex tasks. Examples of such systems are humanoid robots [4, 5, 6], unmanned aerial vehicles (UAVs) with manipulation capabilities [7], mobile manipulators [8] and autonomous underwater vehicle manipulator systems [9, 10].

In the area cooperative robotics in smart factory scenarios, different works have been presented dealing with robot cooperation and human robot collaboration. X-act [11], involving fixed base dual arm robot assembly operations, is one of the most advanced projects from the technological point of view, provides a great insight on the possibilities of technology integration and smart program interfaces. The research carried out focuses on enhancing dual arm robots for making them applicable in a manufacturing environment with fence-less human robot supervision system. The AutoRecon project [12] focuses more on reconfigurability aspects for assembly operations to overcome the limits of the classical fixed linear sequences of operations and to improve fault tolerance, not directly dealing with advanced robot cooperation. The Valeri project [13], also deals with reconfigurable plants, in a particular scenario such as relatively small mobile manipulators for aerospace production, with low programming times leading to increases in both speed and flexibility. The Stamina project [14] aiming to develop a system comprising a fleet of mobile autonomous industrial robots (mobile base manipulators) focusing on De-palletizing, Bin-Picking and Kitting operation, but independently one from each other.

In the last few years, cooperative manipulation and transportation has become

one of the most important research topics on multi-robot systems [15]. The early works on cooperative transportation can be traced back to the seminal work [16], in which a team of mobile robots push an object cooperatively toward a desired position. The authors in [17] follow a similar approach, where the mobile robots are around the object, and focus on the problem of motion planning, dividing the planning problems in a global path planner and a local manipulation one. However, manipulation is performed only by changing the height of the end-effector, i.e. only 1 degree of freedom (DOF). A similar approach is presented in [18].

Following the trend of the aforementioned works, [19] proposes a leader-follower approach, in which the object is assumed the virtual leader; the robots carrying the object are considered followers. The work [20] proposes a leader/follower scheme, where the leader (either a robot or human) can guide the whole group towards the destination by applying a relatively small force, whose effect is amplified by the follower robots as they align their forces with the leader's one. The authors of [21] propose a feedback control law for a cooperative transportation system comprising two car-like vehicles, by assuming that the object is held in two manipulation points, which are revolute joints coupling the two vehicles to a carrier. In none of the previously mentioned works the transportation exploits manipulators, which are needed to pick-up and release objects that are outside the reach of the mobile base. The presence of an actual arm increases the number of DOF to be controlled significantly, increasing the possibilities of manipulation but also the complexity of the control.

Seminal work on cooperative mobile manipulation was presented in [22], using the experimental platform detailed in [23]. The mobile base was dynamically coordinated with the arm using a potential function aimed at maintaining the manipulator joint positions close to their midrange. In [24] a centralized robust adaptive controller is introduced to guarantee that the motion and force trajectories of the constrained object converge to the desired manifolds. However, their approach is limited to a 3 DOF system and neglects all the other control objectives of the system. The work [8] tackles the problem of coordinating two mobile manipulators, employing a task priority inversion technique to increase the priority of safety tasks whenever necessary. In [25] a decentralized task space adaptive control law of cooperative mobile arms is proposed. However, their proposed technique puts safety tasks, such as joint limits, at a lower priority with respect to transporting the object, which implies that they might not be satisfied all the times.

Another approach is presented in [26], where two dual-arm mobile manipulators are employed for solving a cooperative transportation task. In their work, the authors employ a strategy similar to the one first proposed in [22], decoupling the manipulator motion and the mobile platform motion kinematically in task-space. The rationale is that disturbances of the mobile platform such as delayed or imprecise wheel actuation would propagate through the coupled kinematics immediately to the end effectors of both manipulators, if the Jacobian of the whole body is employed to control the end-effectors. When decoupling the mobile platform from the manipulators, disturbances are efficiently compensated before propagating to end effector level. However, the work does not tackle additional control objectives such as joint limits or obstacle avoidance. The same authors have also investigated the model for the internal forces and interaction dynamics in [27, 28].

These projects are very inspiring and challenging, yet many solutions seem to be very task oriented and even if many problems have been already separately tackled, there seems to be a lack of research in the combination of real-time (RT) reactive strategies for the problems that can arise in a complex and mixed environment such as an industrial plant, namely: robot-robot physical cooperation, object force control, obstacle avoidance and low level reactive path control.

Given the high versatility and the fact that such systems need to complete different goals, it became important to have a simple and effective way to specify the control objectives.

Nowadays, the existence of task priority frameworks (see for example [29, 30, 31]) allows putting control objectives in the correct priority order, i.e. safety tasks such as avoiding joint limits or obstacles at highest priority, since they can be deactivated whenever not necessary.

Based on the aforementioned task priority frameworks, several interesting results have been proposed in the literature of underwater and aerial autonomous vehicles. In fact, based on the framework [31], a coordinated control strategy for a single underwater vehicle manipulator system has been presented in [32], with experimental results on floating manipulation, taking into account multiple control objectives at the same time. Based on the same framework, a coordination technique for cooperative transportation was proposed for underwater systems [33], with the specific goal of minimizing the amount of information exchange between the robots due to the severe limitations of acoustic communication channels. Due to the complexity of the systems involved, only simulation results have been pre-

sented.

Another interesting approach is proposed for the coordination of multiple unmanned aerial vehicles for the execution of a cooperative manipulation task [34]. In such a work, the authors employ the null-space behavioural (NSB) control paradigm [35] to control each single agent [36]. In the NSB approach, simple control behaviours are aggregated to form a set of compound behaviours. Since basic behaviours may conflict with each other, and since the priority order is fixed, the approach in [36, 34] builds a set of compound behaviours, where the conflicting basic behaviours have different reciprocal priority. A finite state machine is used to smoothly switch between compound behaviours, depending on which safety objective is about to be violated. However, as the number of conflicting basic behaviours increases, so does the number of compound ones. Conversely, a single hierarchy of tasks/basic behaviour is used in [32], and each single basic behaviour/task is enabled/disabled, with the mechanism as explained in [31].

1.2 Scientific and Technological Objectives

Given this wide spectrum of on-going research, the possibility of having a unifying framework, which aims at providing a control strategy that encompasses most the possible cases, is of great interest. By exploiting the mathematical tools made available by the afore-mentioned task priority framework, the main objective of this PhD work has been to build a robotic control framework on top of it, for controlling all the major robotics structures and their compositions as in figure 1.2. In particular the objective has been to provide these structures with a set of basic capabilities such as (but not limited to): obstacle avoidance, force control, vehicle-arm coordination, multi-robot coordination and context-based control via the definition of so the called *actions* (which will be introduced shortly). To this regard, we need to introduce firstly some conceptual definitions.

An *Individual Robotic Agent* (for short an Agent) is intended to be any robotic structure having its coordinated motion activities governed by an associated, suitably designed, Centralized Agent-control System (for short Agent-Controller, A_C in the sub-figures of 1.2).

A *Multi Robotic Agent* (for short a Multi-agent) is instead intended as the aggregation, or collection, of different Agents, which results when more-than-one of these lasts are all together employed for executing, cooperative activities; via

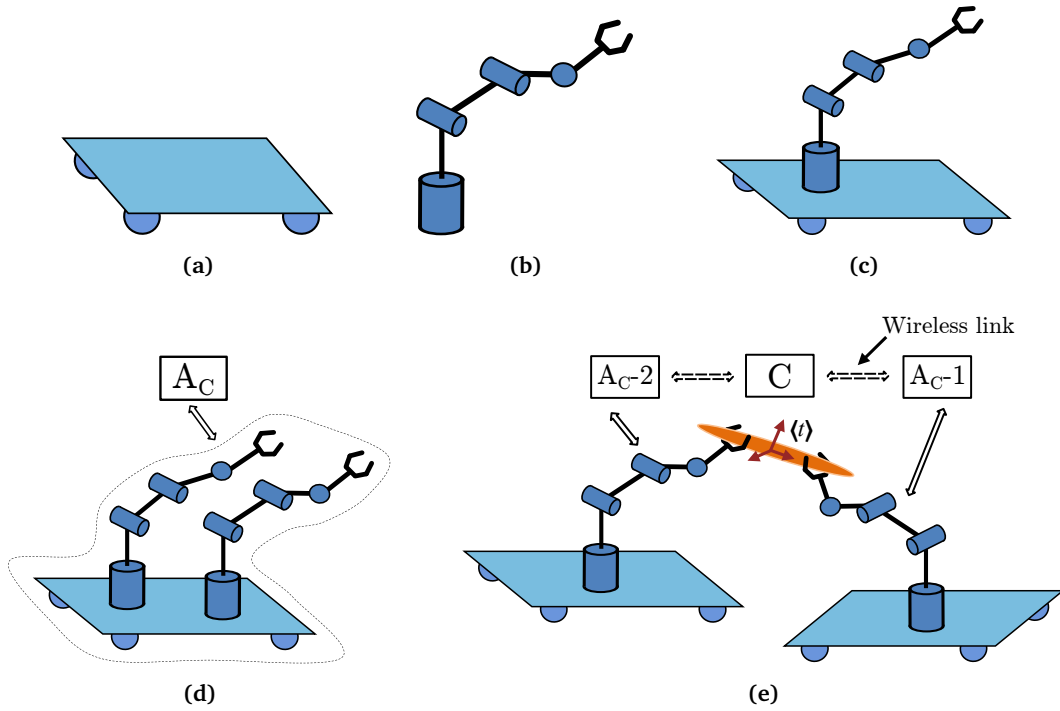


Figure 1.2: Basic structural robotic elements in figures 1.2a and 1.2b, and some of their possible compositions. In figures 1.2d and 1.2e is highlighted a Single-agent vs. Multi-agent control.

the aid of a suitably implemented Coordination Policy, (for short, C in figure 1.2) established among their different agent-controllers. This in turn is supported by a real-time information exchanges with them. Moreover, provided possible, such information exchange should remain confined within the set of variables strictly needed for representing the evolution of the required cooperative activity (i.e. by as much as possible excluding details regarding the very many internal variable supporting the mechanical motion of each involved agent).

The difference between the two concepts can be easily appreciated by comparing the potential performances for instance achievable by vehicle/dual arm system centralized governed as in Figure 1.2d, versus those instead achievable when the same robotic system is considered as a Multi-agent decentralized controlled as in Figure 1.2e. In the second case in fact, just due to the assumed non-complete availability of the merged individual information sets, we have that its potentially expressible performances cannot ever be better than those relevant to its centralized counterpart. The convenience in resorting to decentralized solutions as in Figure 1.2e, might however be appreciated within more advanced scenarios requiring for reconfigurable robotic structures, where these last can transform their mechanical

structure.

What is worth to be developed, is the problem related with the definition of the coordination policies. Policies that must be adopted where a Multi-agent organization de-facto arises in consequence of cooperation activities to be established among robotic structures, not allowed for cabled-data-exchange among them. This is for instance the case of the “occasional” cooperation that could be required to a couple of separate vehicle/one-arm systems (for instance passing an object grasped by one of them, to the other; or manipulating a shared grasped object; or even possibly performing on-fly assembly tasks) where the reduced-bandwidth of a wireless data-exchange, certainly prevents the intensive rates that would instead be necessary for establishing a centralized Agent-Controller also for such resulting “occasional” situations.

The above objectives are important not only for industrial robots, but in general for many complex robotic systems. Indeed, there is a well-established line of research at the GRAAL Lab¹ on the topic of controlling systems with high number of degrees of freedom, such as underwater vehicle manipulator systems. For this reasons, part of the effort has been put in transferring and extending the know-how of the laboratory in underwater robotics to industrial-like scenarios. In fact, stemming from the already established research line on underwater manipulation, the main focus of this PhD has been to extend those results and provide a unifying architecture, at kinematic level, for underwater and ground systems, encompassing single and dual arm robots, with fixed, mobile or floating bases. To meet the aforementioned flexibility goals, as part of this work, a three layer architecture has been investigated, composed by the following three main blocks: the High Level layer (HLL), the Kinematic Control Layer (KCL) and the Dynamic Control Layer (DCL). The main focus of the investigation has been the kinematic layer. The KCL deals with geometric constraints and control objectives that are independent from the underlying dynamic control capabilities, due to the aforementioned layered architecture. The HLL instead is considered as the input provider for our system, and its composition may vary depending on the specific application. The DCL, being tightly related with the underlying hardware, will be considered as given and performing appropriately with respect to the real-time system velocity reference provided by the kinematic layer (also, it is usually not accessible due to proprietary restrictions).

The analysis of the required capabilities for manipulator systems and mobile/float-

¹Genoa Robotics And Automation Laboratory (DIBRIS), *University of Genoa*

ing manipulators has underlined the need of a series of additions and extensions to the already available control framework.

One of the first needs that was highlighted during the analysis was the possibility of performing manipulation using dual arm setups. Dual arm systems have become more and more popular lately. Apart from being a more natural interface for human co-workers, they mostly provide much more manipulation possibilities. Dual-arm application in fact span from assembly and knitting automation task, to human-robot collaborative ones. This work has been focused on defining a particular task than enables the integration of rigid object manipulation in the task priority framework.

Another important objective in robotics is to provide the manipulator with reactive control behaviours. This is of particular importance when the robotic structures shares its operational space with other entities, but it is important in general when a robot has to safely operating in a cluttered environment, which is often the case. As an example, the use of such a framework allows to encompass control tasks such as obstacle avoidance (for example with workers sharing the workstation) or low level reactive path control (rather than specifying the exact path but giving just a high level goal and computing online the elementary steps).

Furthermore, another essential capability that was identified was how to manage the interaction with the environment, at kinematic level. Control techniques that deal with force regulation or safe interaction with the environment do already exists, but are implemented at dynamic level. The problem becomes that of properly integrating these capabilities at kinematic level, and assessing the related performances. In this way, they could be integrated with the rest of the task priority framework, allowing for interaction tasks to be specified in the same way as all the other ones. This is especially relevant when a particular robotic system cannot be controlled at dynamic level, which is the case for many commercial systems.

As instead the use of cooperative robots is concerned, they can be very useful for executing tasks that cannot be carried out by a single agent, as for example handling (i.e. transporting) big objects. Taking into account that each agent exploits the aforementioned single-agent control framework, the problem becomes that of finding a way to coordinate them for the execution of the mission. In the specific case of cooperative transportation, the problem is made even harder by the strict physical interaction occurring between the two robots via the common object, rigidly grasped. Furthermore, especially for underwater environments, the

communication between the robots can be extremely difficult, making the considered problem very challenging.

Last but not least, to cope with complex robotic manipulation scenarios, it has been of particular importance the definition of context-based control in form of what we will call control *actions*. A control action is essentially a set of control objectives that are relevant to achieve an overall, complex behaviour of the system (e.g. grasping an object, moving the base to a goal position, exerting a force on a surface, etc.) . This addition allows to define when a particular control task should be smoothly activated or deactivated. This also allows for reducing the amount of information/detail that are exposed to higher level planning systems, simplifying the efforts of integrating high level control and artificial intelligence techniques.

Summing up, the main contributions related to the scientific objectives of the PhD, have been to develop a generalized control framework capable of:

- C_1 Handling different robotic structures (fixed base arms, single and dual arm mobile manipulators either working on ground or underwater) in a uniform way
- C_2 Integrate a reactive behaviour in the control loop
- C_3 Enable dual-arm manipulation for rigidly-grasped objects
- C_4 Being able to perform interaction tasks with the environment and human beings
- C_5 Handling cooperative transportation between different robots
- C_6 Exposing interfaces for the integration of the KCL with the HLL to show how the KCL can deal with local problems autonomously (e.g. avoiding an obstacle reactively)

Since the task priority framework works at kinematic level, it can be used on both underwater and ground robotic systems, achieving the required re-usability. Moreover, the robot cooperation could introduce new methodologies of manufacturing by avoiding the need for big and costly robots in favour of more economic and reconfigurable solutions. The integration with a force based control enables the possibility of performing tasks on more delicate objects and surfaces by ensuring a maximum threshold for the stress applied by the manipulators, broadening the set

of possible applications.

In the process of achieving these objectives, it has always been of uttermost relevance to keep as much as possible uniformity between marine and industrial scenarios, in order to avoid designing control laws that are relevant only for a very specific robot or situation. From the technological point of view in fact, a key objective has been to develop the software architecture, supporting the control framework, in order to achieve the highest possible re-usability and flexibility. This allowed an easy application of the control framework to different robotic setups, minimizing the development effort to adapt not only the control laws, but also all the collateral supporting software needed to interact with the specific robot-software environment. Indeed, this work is framed within an on-going integration effort between AI techniques and control [37], with the goal of moving towards a smoother combination of the two that could possibly make the difference in terms of performance, flexibility and dependability, especially for factory of the future environments. As a long-term goal, based also on my past interdisciplinary experience in the field of robotics [38, 39], in future work the idea is to enable rich cooperation between humans and a heterogeneous team of robots.

1.3 Experimental Outline

To achieve the scientific objective highlighted in the previous section, of creating a flexible control framework able to encompass different robotic systems in a seamless way, my investigation proceeded by considering setups of increasing complexity: single arm manipulators, fixed based dual arm system and mobile manipulators. Only successively, the problem of coordinating two mobile manipulators systems has been addressed.

An overview of the successive steps of this PhD work is briefly outlined as follows (refer to the next section for a more detailed discussion):

- The control actions have been defined and successfully tested in a complex simulation [40], [41].
- Results on force regulation at kinematic level have been produced in a simulation environment [42].

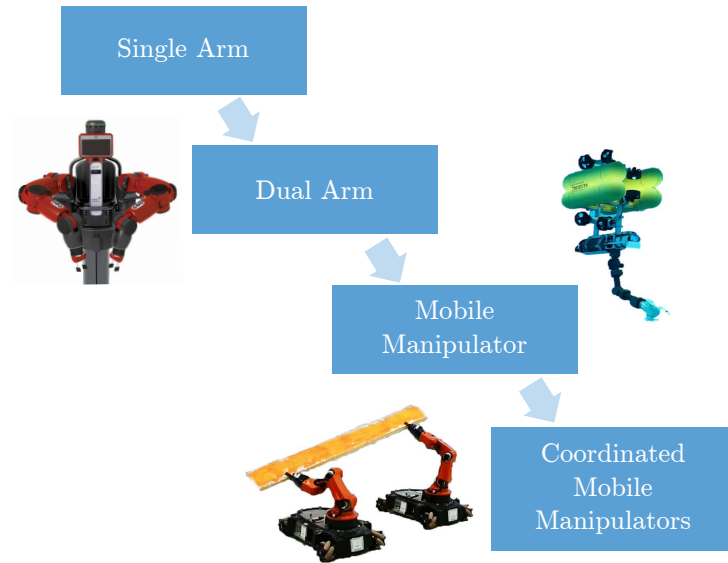


Figure 1.3: General overview of the robot architectures that have been successively investigated.

- Rigid grasp constraint task has been successfully integrated and tested on the Baxter, allowing the use of the Baxter robot for dual arm operations. Obstacle avoidance tasks integrated with the rigid-grasp constraint has been successfully integrated and tested on the Baxter in a human-robot collaboration scenario.
- Preliminary cooperation algorithms for transportation have been developed and simulated using two underwater manipulator systems.
- The cooperative techniques for transportation have been successfully developed and implemented on a pair YouBot robots.

In parallel to the theoretical work, a critical part has been the software definition and implementation for the real-time control architecture. The work started from the previous one developed during the TRIDENT project. However, its limitations and the need to use a new, recently developed real-time robotic middle-ware, have required a new implementation. Given these premises, the new architecture had the important goal to further improve the reuse of each software components. Toward that end, the followed methodology has been to subdivide the architecture in different processes, each one with a specific concern (e.g. Driver, Controller, Communication, Logger). The definition of a standardized interface for such a driver process allows the controller to be as much as possible generic and reusable for different robotised setup. The implementation of the RT framework has been done on

the robotic middle-ware developed at GRAAL (the Ortos RT Framework). However, to facilitate the interaction with other software components that do not depend on real-time requirements, a ROS interface was produced to act as a bridge between the RT and non-RT environments.

To this aim, the developed solutions have been tested in different simulation environments and then on two real robotic structures: firstly, on a bimanual fixed-base Baxter robot and later on a pair of KUKA YouBots mobile manipulators. The results achieved in the single agent scenarios show the capabilities of the underlying control architecture in solving local problems, in the perspective of simplifying the amount of work of higher planning levels that can therefore focus better on high level objectives. The successive experiments performed with the multi-agent system show the ability of the framework to deal with mobile base manipulators and its ability to carry out cooperative operations tasks.

The thesis is organized as follows. Chapter 2 recalls the theory behind the task priority control framework, defining all the basis upon which the thesis work is built upon. In Chapter 3 is highlighted the flexibility of the proposed architecture in tackling many different kind of robotic applications. After, an extension for a cooperative multi robot scenario is presented in Chapter 4, and the relative results obtained in simulation and then experimentally have been presented in Chapter 5. The second-last chapter is dedicated to briefly describe the software modules developed for all the experiments carried out. Finally, some conclusions are given in Chapter 5.2.1.

SINGLE AGENT CONTROL: THEORY

As anticipated in the introduction the control framework presented in this thesis is based on the cascade of blocks shown in Fig. 2.1. In particular, the architecture is constituted by three main general blocks:

1. The Mission Manager is in charge of supervising the execution of the current *mission*, and generates the corresponding *actions* to be executed by the Kinematic Control Layer. As it will be explained later in section 2.4, an action is any prioritized list of control objectives to be concurrently achieved, and a mission is a sequence (or graph) of actions.
2. The Kinematic Control Layer (KCL) implements the proposed task priority control framework, and 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 references for the vehicle and the manipulator.

The thesis analysis focuses on the Kinematic Control Layer, since it is the one implementing the proposed task priority approach. The interfaces with the higher level (Mission Manager) and the lower one (DCL) are highlighted whenever relevant. By the definition we have given for the Mission Manager follows that it will

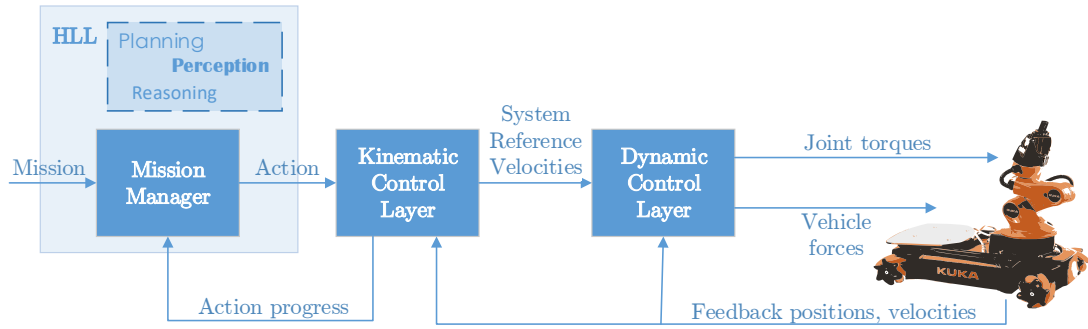


Figure 2.1: The overall architecture: the Kinematic Control Layer is the one implementing the proposed task priority approach, executing the current action scheduled by a Mission Manager; the output of the Kinematic Control Layer are the system velocities, tracked by the underlying Dynamic Control Layer.

be application dependent. In general it is the higher level layer in a given software architecture. Therefore we may also refer to it as the HLL (High Level Layer).

The unifying KCL control architecture presented can be applied both in the case of partial (assisted) teleoperation and fully autonomous operation. The main features of this architecture can be summarized as:

- F_1 inequality control objectives are handled without over-constraining the system, allowing them to be placed at higher priority whenever necessary, e.g. safety control objectives can have high priority (section 2.5);
- F_2 switching between actions (an action is a set of prioritized control tasks) can be done easily, and allows the higher planning levels to interface with the control one in terms of such building blocks, which cover a great deal of underlying control details (section 2.5.1).
- F_3 the presence of vehicle underactuations can be managed, to the best extent possible, with the addition of a suitable highest priority non-reactive control task (section 2.7);
- F_4 vehicle inaccuracies in tracking the desired velocity are not propagated to the end-effector thanks to a vehicle-arm coordination scheme, based on two parallel optimizations (section 2.8);
- F_5 the vehicle-arm coordination scheme naturally allows a multi rate control of the two subsystems (section 2.9);

All these features directly impact on our control architecture, enlarging the span of applications that a robotic controller built on top of it can deal with, thus enforcing contribution C_1 .

2.1 General Definitions

Before starting the discussion we first introduce some notation in the next subsection. Then some definitions are given in the successive subsection, while some preliminaries on the regularized pseudo inverses are given in the last one.

2.1.1 Notation

Vectors and matrices are expressed with a bold face character, such as $\mathbf{M}$, whereas scalar values are represented with a normal font such as γ . Given a matrix $\mathbf{M}$:

- $M_{(i,j)}$ indicates the element of $\mathbf{M}$ at the i -th row and j -th column;
- $\mathbf{M}^\#$ is the exact generalized pseudo-inverse, i.e. the pseudo inverse of $\mathbf{M}$ performed without any regularizations.

Further, less used notation is introduced as needed.

2.2 Definitions

Let us introduce some definitions that will be used thoroughly in this paper. Here we will consider the general case of a floating dual-arm manipulator, which encompasses all the type of robots, then every other system considered successively will lie in a subspace derived by an appropriate downgrade of this general case. Let us define:

- The system configuration vector $\mathbf{c} \in \mathbb{R}^n$ of a generic mobile manipulator as

$$\mathbf{c} \triangleq \begin{bmatrix} \boldsymbol{\eta} \\ \mathbf{q}_L \\ \mathbf{q}_R \end{bmatrix}, \quad (2.1)$$

where $\boldsymbol{\eta} \in \mathbb{R}^m$ is the vehicle *generalized coordinate position vector* and $\mathbf{q}_L \in \mathbb{R}^l$ and $\mathbf{q}_R \in \mathbb{R}^r$ are the left and right arm configuration vectors, supposing that that the left arm has l DOF and the right one has r DOF. In particular $\boldsymbol{\eta}$ is

the stacked vector of the position vector η_1 , with components on the inertial frame $\langle w \rangle$, and the orientation vector η_2 , the latter expressed in terms of the three angles yaw, pitch and roll (applied in this sequence) [43]. From the above definitions it results $n = l + r + m$;

- The system velocity vector $\dot{y} \in \mathbb{R}^n$ of the mobile manipulator as

$$\dot{y} \triangleq \begin{bmatrix} \mathbf{v} \\ \dot{\mathbf{q}}_L \\ \dot{\mathbf{q}}_R \end{bmatrix}, \quad (2.2)$$

where $\mathbf{v} \in \mathbb{R}^m$ is the stacked vector of the vehicle linear velocity vector $\mathbf{v}_1$ and the vehicle angular velocity vector $\mathbf{v}_2$, both with components on the vehicle frame $\langle v \rangle$. While $\dot{\mathbf{q}}_L \in \mathbb{R}^l$ and $\dot{\mathbf{q}}_R \in \mathbb{R}^r$ are the left and right arm joint velocities. For the time being, this work will be developed with the assumption of a vehicle fully actuated in its 6 degrees of freedom, therefore $m = 6$, and the system velocity vector will coincide with the control vector. Section 2.7 will show how to relax this hypothesis.

2.3 Control Objectives

Let us start the discussion by considering what the robotic system needs to achieve. The goals of the system might include, for example, maintaining a certain distance from another vehicle, reaching a waypoint, or avoiding the arm joint limits. Mathematically speaking, let us consider a scalar variable $x(\mathbf{c})$, i.e. a variable dependent on the current configuration of the robot and the environment. Then, for this variable, two broad classes of objectives can be defined:

1. The requirement, for $t \rightarrow \infty$, that $x(\mathbf{c}) = x_0$ is called a *scalar equality control objective*.
2. The requirement, for $t \rightarrow \infty$, that $x(\mathbf{c}) < x_M$ or $x(\mathbf{c}) > x_m$ is called a *scalar inequality control objective*.

The mathematical symbols ‘=’ and ‘ $\leq$ ’ will be used throughout the thesis to identify equality and inequality scalar objectives respectively.

The use of scalar objectives is to simplify the presentation of the approach. However, let us remark that it is not a limitation of the approach. In fact, consider a

vector $\mathbf{h} \in \mathbb{R}^m$, and consider m different scalar objectives, one for each of its components. Then, all together the m objectives allow to define a requirement for the whole vector $\mathbf{h}$, for example having some of its components within some ranges, while the others attaining certain values.

Other than the specific class, the control objectives can be divided in different categories, depending on their purpose and, as it will be clear later, on their priority. In fact, control objectives can be classified as:

- C physical **constraints** objectives, i.e. interacting with the environment;
- S system **safety** objectives, e.g. avoiding joint limits or obstacles;
- P objectives that are a **prerequisite** for accomplishing the mission, e.g. maintaining the manipulated object in the camera angle of view;
- A **action** defining objectives, i.e. what the system really needs to execute to accomplish the current user defined action;
- O **optimization** objectives, i.e. objectives that do not influence the mission, but allow to choose between multiple solutions, if they exists.

The capitalized letters associated with the class types will be used throughout the thesis to identify a specific type of objective (the Nomenclature section at the beginning can be used as a reference).

2.3.1 Reactive Control Tasks

To each scalar control objectives there is always associated a *feedback reference rate* $\dot{\hat{x}}$, whose role is to drive the associated variable $x(\mathbf{c})$ toward an arbitrary point x^* , located inside the interval where the objective is satisfied. For instance, an example of feedback reference rate is the following one:

$$\dot{\hat{x}}(x) \triangleq \gamma(x^* - x), \gamma > 0, \quad (2.3)$$

where γ is a positive gain proportional to the desired convergence rate for the considered variable.

For each of the considered variables $x(\mathbf{c})$, the Jacobian relationship

$$\dot{x} = J\dot{y} \quad (2.4)$$

links the rate of change of the variable itself with the system velocity vector $\dot{\mathbf{y}}$ through the vector $\mathbf{J} \in \mathbb{R}^n$. So, having the actual $\dot{\mathbf{x}}$ as much as possible equal to the desired $\dot{\hat{\mathbf{x}}}$ is termed a *reactive control task*. The letter ‘r’ will identify reactive tasks throughout the thesis.

2.3.2 Non-reactive Control Tasks

In the previous sections, the triple control objective, reactive control task and activation function has been defined. However, there are cases where a control task is defined directly in a certain task velocity space, without having an associated control objective. As an example, consider the case where a human operator wants to control the end-effector by generating velocity references through a joystick. In such a case, there is no control objective associated with the control task, because there is not a position to reach. In fact, the reference rate is generated by the user, rather than being the output of a feedback control loop as in (2.3). The letters pair ‘nr’ will identify non-reactive tasks throughout the thesis.

2.4 Actions and Missions

In the terminology of this work, an *action* is a prioritized list of control objectives and associated reactive tasks, with the possible addition of any non-reactive tasks, to be *concurrently* managed. For example, for a robotic manipulator, a manipulation action could be represented by following list of objectives

- $\mathbf{o}_1)$ [r, $\leq$, S] Arm joints limits
- $\mathbf{o}_2)$ [r, $\leq$, S] Arm manipulability
- $\mathbf{o}_3)$ [r, =, A] Tool frame position control
- $\mathbf{o}_4)$ [r, =, A] Tool frame orientation control
- $\mathbf{o}_5)$ [r, $\leq$, 0] Arm preferred shape

where it is natural to see *safety* objectives (S) such as joint limits at the highest priority. It is followed by the manipulability objective that allows for a better execution of the overall position objective, since it avoids the problems of high joint velocities occurring near kinematic singularities. Finally, an optimization objective such as the preferred shape can be defined, however its priority will be lower than the position control, to avoid over-constraining the system to a specific configuration. See Section 2.3 for more insight on task types. A more in-depth explanation

for the single tasks will be introduced in the application section, every time a new control objective is introduced.

Further examples will be given in Chapter 3. For the time being, the following aspects can be highlighted:

- In principle, within an action, there could be an arbitrary number of prioritized objectives, organized in different ways. However, following the categories of objectives presented in section 2.3, usually the main difference between actions will be the objective(s) needed to achieve its particular ultimate goal and possibly some other prerequisite ones, while the safety tasks will be the same.
- Actions can be temporally sequenced accordingly with a *mission* plan represented by a decisional action graph, having the actions as nodes and the arcs as logic decision alternatives, with just a few parameters relevant to the action ultimate-goal only.
- The aforementioned two points allow to highlight how the interface of the KCL with the Mission Manager should therefore result simplified, allowing an easier implementation of the latter.
- Moreover, transitions from an action to another one located at the end of a selected arc, should be smoothly activated.

2.5 Activation Functions

For every control objective and relative reactive control task, there is always an associated activation function to be considered. The role of the activation function, as the name implies, is to specify whenever the control objective is relevant for the system. For example:

- Maintaining the object to be grasped within the camera system cone of vision is relevant only when the object is close to the boundaries, since there is no need to have it *exactly* centred;
- Maintaining a joint away from its limit is relevant only in the vicinity of the limit itself, but it *should not overconstrain* the system when the joint is sufficiently far away.

Motivated by the above considerations, let us define an activation function so composed:

$$a(x) = a^i(x), \quad (2.5)$$

where $a^i(x) \in [0, 1]$ is continuous sigmoidal function of the objective variable x , which assumes zero values within the validity region of the associated objective. For example, assuming an objective of the class $x(c) > x_{min}$:

$$a^i(x) = \begin{cases} 1, & x(c) < x_{min} \\ s(x), & x_{min} < x(c) < x_{min} + \Delta \\ 0, & x(c) > x_{min} + \Delta \end{cases} \quad (2.6)$$

where $s(x)$ is any continuous function joining the two extrema, and Δ is a value that creates a buffer zone, where the inequality is already satisfied, but the activation value is still greater than zero, preventing chattering problems around the inequality control objective threshold x_{min} . An example of sigmoidal activation function is reported in Fig. 2.2b. Throughout the thesis we will make use of the sigmoidal function so it's useful to define a compact notation to identify different types of sigmoids. Having $[x_{min}, x_{MAX}]$ the two minimum and maximum extrema for the sigmoid curve on the x axis, and $[0, 1]$ the two extrema for the sigmoid curve on the y axis (the activation value), we can define:

- An increasing sigmoidal activation function as

$$\sigma_{\uparrow}(x_{min}, x_{MAX}) \quad (2.7)$$

- A decreasing sigmoidal activation function as

$$\sigma_{\downarrow}(x_{min}, x_{MAX}) \quad (2.8)$$

In general, for multidimensional control objective the activation function for a particular task k will be a matrix of this type:

$$\mathbf{A}_k = \begin{bmatrix} a_{k,1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & a_{k,l} \end{bmatrix}, \quad (2.9)$$

where here $a_{k,1}$ is the i -th element on the trace of $\mathbf{A}$ for the k -th task.

For non-reactive control tasks, since the variable $x(c)$ is not defined, the activation function (2.5) reduces to be $a(x) = a^i(x) \equiv 1$ as in Figure 2.2a. From now on, the distinction between reactive and non-reactive control tasks will be dropped, and the generic term control task will be used, unless otherwise specified.

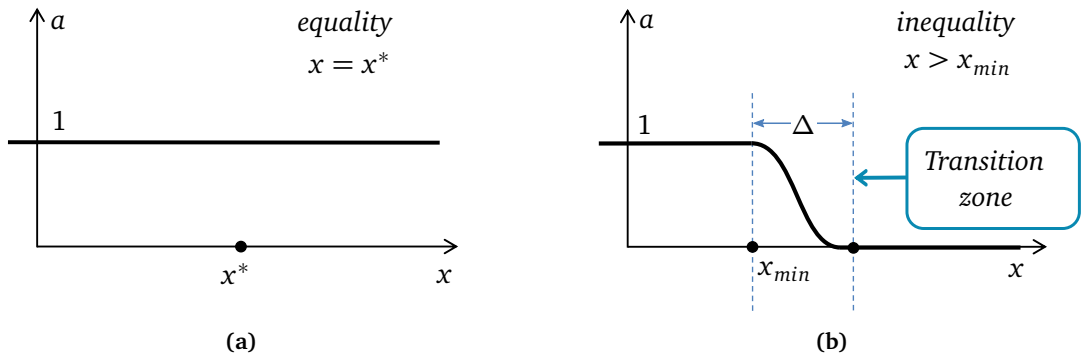


Figure 2.2: Comparison of task activations for equality and inequality control objectives. In figure 2.2b and example of $\sigma_{\downarrow}(x_{min}, x_{min} + \Delta)$ decreasing sigmoidal function.

2.5.1 Action Transition

The activation and deactivation of a task, apart from avoiding unnecessary system constraints, can serve another important purpose. In fact, it can be exploited for providing a smooth transition between actions, so to satisfy the requirements highlighted in 2.4. Let us consider the following two actions $\mathcal{A}_1$ and $\mathcal{A}_2$, composed by objectives abstractly labelled with alphabetic letters, where $A \prec B$ denotes that A has higher priority than B :

$$\mathcal{A}_1 : A \prec B, C, D$$

$$\mathcal{A}_2 : A \prec D \prec C, E$$

where A, C, D are in common, but with D at a different priority ordering w.r.t. C . Now consider the following merged list:

$$\mathcal{U} : A \prec D_1 \prec B, C, D_2, E;$$

where D_1 and D_2 represent the same objective D , but with a different priority. It is clear that, through insertion/deletion of some of the entries, the two original lists can be reconstructed. For example, $\mathcal{A}_1$ can be obtained by removing D_1 and E from the unified list. Conversely, $\mathcal{A}_2$ can be obtained by removing the control objectives D_2 and E .

To make possible this transitions when switching between actions, the activation function (2.5) is modified to become:

$$a(x, \mathbf{p}) = a^i(x) a^p(\mathbf{p}), \quad (2.10)$$

where $a^p(\mathbf{p}) \in [0, 1]$ is a continuous sigmoidal function of a vector of parameters $\mathbf{p}$ external to the control task itself. In particular, the $a^p(\mathbf{p})$ can be conveniently parametrized by the previous and current action being executed, and the

time elapsed in the current action, to obtain the desired activation/deactivation smooth transition. Of course, this new activation function $a^p(\mathbf{p})$ applies to both to reactive and non-reactive control tasks since it does not depend on the variable $x(\mathbf{c})$. Having N tasks, identified by their k index, and two successive actions $\mathcal{A}_1$ and $\mathcal{A}_2$, we can define the following algorithm for smoothly transition between actions:

Algorithm 1 Tasks transition algorithm

```

1: for  $k \leftarrow 1$  to  $N$  do
2:   if  $\text{task}_k \in \mathcal{A}_1 \wedge \text{task}_k \in \mathcal{A}_2$  then
3:      $a^p(\mathbf{p}) = 1$ 
4:   end if
5:   if  $\text{task}_k \in \mathcal{A}_1 \wedge \text{task}_k \notin \mathcal{A}_2$  then
6:      $a^p(\mathbf{p}) = \text{decreasingSigmoid}(t)$ 
7:   end if
8:   if  $\text{task}_k \notin \mathcal{A}_1 \wedge \text{task}_k \in \mathcal{A}_2$  then
9:      $a^p(\mathbf{p}) = \text{increasingSigmoid}(t)$ 
10:  end if
11: end for
    
```

Once all the actions that the system can execute have been defined, the unified list can be constructed. Many safety tasks will be common to all actions, and they will be at the same (high) priority level in each of them, therefore for such tasks it will result that $a^p(\mathbf{p}) \equiv 1$. Conversely, $a^p(\mathbf{p})$ will be mostly exploited to activate/deactivate action-defining objectives and prerequisite ones, calculated as in Algorithm 1.

2.6 Task Priority Inverse Kinematics

Given an action $\mathcal{A}$, for each priority level k , the following matrices and vectors are consequently defined:

- $\dot{\mathbf{x}}_k \triangleq [\dot{x}_{1,k} \ \cdots \ \dot{x}_{m,k}]^T$ is the stacked vector of all the reference rates of the scalar control tasks;
- $\mathbf{J}_k$ is the Jacobian relationship expressing the current rate-of-change of the k -th task vector $[\dot{x}_{1,k} \ \cdots \ \dot{x}_{m,k}]^T$ w.r.t. the system velocity vector $\dot{\mathbf{y}}$, simply resulting from the stacking of the row vector ones in (2.4);
- $\mathbf{A}_k \triangleq \text{diag}(a_{1,k}, \dots, a_{m,k})$ is the diagonal matrix of all the activation functions described in (2.10).

Let us remark how inequalities introduced in section 2.3 are objectives to be eventually reached that, with the adoption of the reference law (2.3), are transformed in control tasks, i.e. velocity tracking minimization problems. However, as already mentioned, to avoid overconstraining the system, they must be deactivated whenever the inequality is satisfied.

To the aim of finding the system velocity reference vector $\dot{\mathbf{y}}$ that satisfies the above expressed priority requirements between the different objectives, a straightforward idea is to employ the diagonal matrix $\mathbf{A}_k$ as a weight in the minimization process to deactivate tasks that are not anymore relevant, thus leading to the following sequence of nested minimization problems, corresponding to the so called Task Priority Inverse Kinematics (TPIK):

$$S_k \triangleq \left\{ \arg \min_{\dot{\mathbf{y}} \in S_{k-1}} \left\| \mathbf{A}_k (\dot{\mathbf{x}}_k - \mathbf{J}_k \dot{\mathbf{y}}) \right\|^2 \right\}, \quad k = 1, 2, \dots, N, \quad (2.11)$$

where S_{k-1} is the manifold of solutions of all the previous tasks in the hierarchy, $S_0 \triangleq \mathbb{R}^n$, and N is the total number of priority levels. Unfortunately, the task priority implementation as suggested above turns out to be affected by discontinuities in $\dot{\mathbf{y}}$ when $\mathbf{A}_k$ is varying some of its entries (i.e. within the so called transition zones) due to the fact that the resulting weighted pseudo-inverse $(\mathbf{A}_k \mathbf{J}_k)^\# \mathbf{A}_k$ is actually invariant to the weights on the rows of $\mathbf{J}_k$ that are linearly independent, as shown in [44].

In order to cope with these invariance problems, a specialized *task oriented regularization* has been introduced in the minimization problem, as explained in [31]. Such a specialized regularization, coupled with the classical singular value oriented one to deal with singularities of the involved matrices, was the novel idea of the proposed new pseudo inversion framework.

The adopted specialized regularization will be now briefly outlined, by now making reference to the following new form of TPIK problem itself

$$S_k \triangleq \left\{ \arg \text{R-} \min_{\dot{\mathbf{y}} \in S_{k-1}} \left\| \mathbf{A}_k (\dot{\mathbf{x}}_k - \mathbf{J}_k \dot{\mathbf{y}}) \right\|^2 \right\}, \quad k = 1, 2, \dots, N, \quad (2.12)$$

where the notation R-min is introduced for underlining that each minimization is performed via the specialized form of [31].

Within such a specialized framework (named *iCAT: inequality Constraints And Task transitions*) the above TPIK problem (2.12) results in the following algorithm, to be initialized with

$$\boldsymbol{\rho}_0 = \mathbf{0}, \quad \mathbf{Q}_0 = \mathbf{I}, \quad (2.13)$$

and then, for $k = 1, \dots, N$

$$\begin{aligned} W_k &= J_k Q_{k-1} (J_k Q_{k-1})^{\#, A_k, Q_{k-1}}, \\ Q_k &= Q_{k-1} (I - (J_k Q_{k-1})^{\#, A_k, I} J_k Q_{k-1}), \\ \rho_k &= \rho_{k-1} + \text{Sat} \left(Q_{k-1} (J_k Q_{k-1})^{\#, A_k, I} W_k (\dot{\mathbf{x}}_k - J_k \rho_{k-1}) \right), \end{aligned} \quad (2.14)$$

where

- the special pseudo inverse operator $(\cdot)^{\#, A, Q}$ is strictly needed to cope the invariance problems mentioned above. Its definition is reported in [31] and will be omitted here.
- ρ_k is the vector of the system control variables, which results constructed at the k -th minimization stage, through the last equation in (2.14);
- $(\dot{\mathbf{x}}_k - J_k \rho_{k-1})$ is the k -th task reference rate minus the task space contribution of the $k-1$ control vector ρ_{k-1} , i.e. the influence exerted on the current k -th task by the presence of all the higher priority tasks;
- Q_{k-1} is a $n \times n$ matrix that takes into account the control directions (totally or partially) used within all the higher priority tasks;
- W_k is a $m \times m$ matrix, where m is the row-dimension of the control task at the current priority level, which is suitably structured as indicated to prevent discontinuities in the control variables occurring between different priority levels;
- The function $\text{Sat}(\cdot)$ implements the management of control variable saturations, accordingly with what suggested in [45].

At the end of the above iterative process, the final system velocity vector is therefore $\dot{\mathbf{y}} = \rho_N$. The interested reader can find all the relevant details of this procedure in [31], where a comparison with other task priority approaches, as for instance those appearing in [46, 29, 30], is given.

2.6.1 Remarks on Conflicting Objectives

Until now, no remarks on the feasibility of a hierarchy of control objectives and corresponding tasks was made. An action is concluded successfully when its *action defining* objectives are satisfied. For example, a grasping action $\mathcal{A}_g$ is concluded when the end-effector is in the desired final position and the gripper has

been closed. However, the action-defining objectives are usually at a low priority in the hierarchy, as they have a lower priority w.r.t. objectives that deal with safety considerations or to objectives that represent necessary preconditions (e.g. centring the camera on the object to be grasped).

Therefore, a necessary condition to guarantee that an action can be terminated is the existence of a *non-void* set of configurations, to be eventually reached by the system, where all high priority and action-defining objectives are *contemporaneously* satisfied. Only control objectives that are lower priority than the action-defining one(s) can remain unsatisfied (e.g. maintaining a preferred shape), since they usually express a *preference*.

The presence of such a *non-conflicting zone* is, however, not a sufficient condition, as it does not imply its reachability in correspondence of all system initial configurations starting outside it. A classical example regards navigation in presence of obstacles, where it is well known that reactive control strategies might get stuck in local minima. Avoidance of the obstacle is guaranteed, but convergence to the final desired position is not. Such a reachability problem affects all the task priority strategies, and in general all reactive control approaches.

In the proposed architecture, this problem needs to be addressed at a higher level, within the Mission Manager block. The creation of different control actions helps in defining a set of well structured blocks, that can be sequenced by the Mission Manager. Furthermore, the KCL should continuously monitor its objectives, and eventually report to the higher level the inability of satisfying its objectives (e.g. since it is stuck in a local minima). At that point, the Mission Manager job would be to devise an alternative solution, e.g. a different sequencing of action or different parameters (i.e. a new path to follow). A detailed discussion of how the Mission Manager could be implemented falls outside the scope of the current work.

2.7 Control in Presence of Vehicle Underactuations

Until now the vehicle was assumed fully actuated in all its degrees of freedoms and it was implied the robot was also able to actuate that degrees of freedom. In many cases, for surface robots this is not the case, and to avoid unnecessary calculation and reduce the computation time we can often reduce the size of the configuration vector to fit only the set of useful variable. For example, as it has been the case for the experiments in section 5.2, the robot base can only move in 3 directions out

of six (x, y, θ_{yaw}) , and all the other feedback variables will be constant and equal to zero. However, when a robot can actually move in a particular direction but it has no control over it, the proposed architecture can also be used for managing underactuated vehicles to the best extent possible.

Consider an example where an underwater vehicle does not have thrusters allowing to control the angular velocities along the vehicle x and y axes (roll and pitch), which is often the case for small size vehicles. Then, consider the following *non-reactive* control task where:

- Its reference rate $\dot{\mathbf{x}}$ is composed by the vector of the *measured* vehicle velocities (all vehicle velocities, including the non actuated ones, are assumed measurable). Therefore, for such a task $\dot{\mathbf{x}} = [\mathbf{v}_1^T \ \omega_x \ \omega_y \ \omega_z]^T$, where $\mathbf{v}_2$ has been explicitly written in terms of its components;
- Its Jacobian is simply $[\mathbf{0}^{6 \times l} \ \mathbf{I}^{6 \times 6}]$, where the first block of zeroes regards the arm joint velocities that are kinematically non-influencing the vehicle ones;
- Its activation function, in the considered roll and pitch underactuation example, is the constant diagonal matrix $\text{diag}(0, 0, 0, 1, 1, 0)$ selecting the non-actuated velocities only.

Then, provided such a task is located at the *top of the hierarchy*, the following facts hold true:

- The vehicle angular velocity components of $\boldsymbol{\rho}_1$, formally evaluated as if all vehicle velocities were actuated, will therefore be returned, by the procedure itself, just coinciding with the measured ones, i.e.

$$\boldsymbol{\rho}_1 = [\mathbf{0}^{1 \times l} \ \mathbf{0}^{1 \times 3} \ \omega_x \ \omega_y \ 0]^T.$$
- The matrix $\mathbf{Q}_1$ will prevent all the lower priority tasks from interfering with the highest priority one, effectively inhibiting them from changing the values of $\boldsymbol{\rho}_k$ in the x and y angular velocities components.
- Each lower priority task will take into account the current angular velocities ω_x and ω_y of the vehicle, during the computation of their task references $(\dot{\mathbf{x}}_k - \mathbf{J}_k \boldsymbol{\rho}_{k-1})$. Therefore, the final control vector is necessarily the best that the procedure can provide, given that some vehicle velocities are non actuated, but measured.

At the end of the procedure, the values of $\dot{\mathbf{y}}$ corresponding to the non-actuated degrees of freedom are then trivially discarded, while the remaining (actuated) ones are instead sent to the DCL to be tracked. In general, with a proper choice of the activation function, the proposed architecture can actually cope with any kind of vehicle underactuation; thus, possibly, even underactuation caused by actuators failures, provided they are detected and the vehicle velocities remain all measurable.

Naturally, the remarks made in section 2.6.1 are more demanding if the vehicle is underactuated. For example, centring a vehicle-fixed camera on the target might be more difficult as arm movements would move the centre of mass and therefore create a different passively stable point in roll and pitch, with possibly negative consequences on target camera centring. Also, the assumed underactuated pitch velocity could obviously create risks of having the pitch angle getting close to its upper limit of 90 degrees.

In any case, for reducing the risks of the above inconveniences, nothing prevents the addition of a further objective aimed at maintaining the overall system centre of mass substantially aligned under the centre of buoyancy, along the *z-axis of the vehicle frame*.

Summing up, the above discussion has briefly outlined how the proposed control architecture is able to manage, to the best extent possible, the existence of vehicle underactuations, even when caused by occasional actuators failures, via a suitable structuring of its objectives and tasks.

2.8 Vehicle-Arm Coordination Scheme

An important feature of the proposed architecture is that the effects of the vehicle velocity tracking inaccuracy are taken into account. In particular, disturbances of the mobile platform such as delayed or imprecise wheel actuation would propagate through the coupled kinematics immediately to the end effector, if the Jacobian of the whole body is employed to control the end-effectors. This for example, is particularly the case for the mobile manipulator we used in our experiments, the YouBot, which is using two pairs of Mecanum wheels [47] to move. Even if they are a clever workaround to the non-holonomy typical of wheeled robots, the odometric measurement is intrinsically difficult due to the passivity of the wheel's surface contact points. Therefore the velocity control of the vehicle results worse than the one

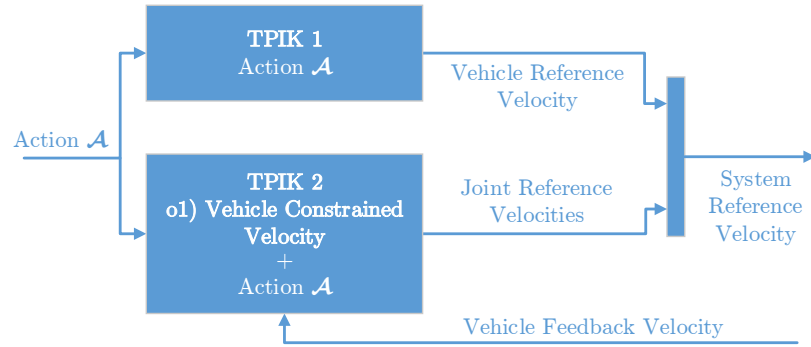


Figure 2.3: A detailed view of the Kinematic Control Layer, with the two Task Priority Inverse Kinematics blocks highlighted, performing the vehicle velocity tracking compensation scheme in presence of an input action $\mathcal{A}$.

for the manipulator's joints. Furthermore, the lower ratios of force/mass for the vehicle than for the manipulator exacerbates this issue.

If manipulator and vehicle motions are not kinematically decoupled, then disturbances of the floating base would propagate through the coupled kinematics immediately to the end effector of the manipulator. The idea of decoupling the two subsystems is well known in ground mobile manipulation, where delayed or imprecise wheel actuation has similar effects to those of the thrusters, and some strategies were proposed in [22] and successively in [26]. However, these approaches are not framed within a task priority approach, and therefore the mobile base does not support the manipulator in achieving all its control objectives such as joint limits or obstacle avoidance.

To cope with this problem, the idea is to have two optimizations (2.14) running in parallel as depicted in Fig. 2.3. Given an action $\mathcal{A}$ to be executed, the parallel scheme which coordinates the vehicle and arm movements works as follows:

1. The first optimization (TPIK 1) considers the vehicle together with the manipulator, as a fully controllable system. The hierarchy of tasks corresponding to action $\mathcal{A}$ is solved as outlined in section 2.6, and of the whole result $\dot{\mathbf{y}}$, only the vehicle reference velocity is used, while the manipulator part is discarded. If the vehicle is underactuated, the additional non-reactive control task presented in section 2.7 is placed at the top of the task hierarchy;
2. The second optimization (TPIK 2) considers the vehicle as *totally non-controllable*. Therefore, the non-reactive control task presented in 2.7 is always used and all the vehicle degrees of freedom are initialized with the corresponding mea-

sured velocities. Therefore, the vehicle constrained velocity task is placed on top of the task hierarchy of action $\mathcal{A}$, as depicted in the TPIK 2 block of Fig. 2.3.

In such an optimization, de-facto only the manipulator variables are subject to be optimized. In fact, the outputs of this procedure are the *optimal* joint velocities in correspondence of the *measured* vehicle velocity.

The two outputs are then forwarded to the corresponding parts of the DCL to be tracked.

Therefore, thanks to the TPIK 2 optimization, the arm joint velocities are always the optimal ones based on the current vehicle velocities, independently of any vehicle inaccuracy in tracking the desired ones generated by TPIK 1. As a remark, note that if the vehicle were tracking its velocities perfectly, the joint velocities output of TPIK 2 would be exactly the same as the one discarded in TPIK 1.

2.9 Vehicle-Arm Multi-rate Control

The parallel strategy for arm-vehicle coordinated control is also suitable for implementing multi-rate control of the two subsystems. For example, let us consider the control rates of a vehicle-manipulator system having a 10 Hz control rate for the vehicle and a 100 Hz rate for the manipulator.

1. At time $t = 0$ s, both TPIK 1 and 2 will be executed, and the corresponding vehicle and joint velocities will be generated and sent to the DCLs.
2. At $t = 0.01$ s, TPIK 2 will run and generate new reference velocities for the manipulator, taking into account any new feedback (joint positions, vehicle velocity, etc.), while the reference velocity for the vehicle will be kept constant. (i.e. the one generated in point 1 of this procedure).
3. The above step is repeated until the period for the vehicle control has elapsed (i.e. 100 ms).
4. At $t = 0.1$ s, both TPIKs will be run and everything will proceed from point 1 again.

SINGLE AGENT CONTROL: APPLICATIONS

As described in the introduction, one of the main objectives of this PhD work is to provide a set of (not so) basic capabilities for mobile manipulator, to serve as building block for more complex tasks. In this chapter we will see the TPIK applied in a series of different scenarios, where the capabilities introduced are usually directly related to the definition of a particular control task. The control framework presented so far has been in fact implemented both in Matlab (for simulation) and on a RT control framework (for experimental testing).

Firstly, in Section 3.1 we will see the task priority algorithm tested on a fixed based manipulator with some simple tasks active.

Then, in Section 3.2 we will see how is possible, implementing a dedicated task, to: perform dual-arm control operations in the case of a rigid bimanual grasp; provide obstacle avoidance capabilities to the manipulator. After validating the proposed strategy experimentally, in a second set of experiments the control framework was tested within a human-robot interaction scenario. The hardware used for these experiments has been a Baxter by Rethink Robotics™, a fixed base bimanual robot.

Finally, in Section 3.3, we will see another extension of the framework, to enable force control regulation tasks at the kinematic level. In this case the study-case involved a UVMS carrying out a pipe inspection where the robot has to maintain a desired contact force between the pipe and the end-effector. For this task, the proposed strategy has been tested in a simulation environment.

3.1 Single-Arm Control

In the perspective of gradually introducing new *capabilities*, in this first experiment we will have a very simple action with a set of basic task that will be always present in our experiments. In this section an experimental trial with a 7 d.o.f. manipulator available in the lab is presented.

For the KCL, recalling the definitions we gave in Section 2.1, having for this setup a single manipulator with 7 DOF, we have $n = l = 7$, and we can downgrade the general case by defining the following control variables:

- The system configuration $\mathbf{c} \in \mathbb{R}^n$ and velocity $\dot{\mathbf{y}} \in \mathbb{R}^n$ vectors of the robot

$$\mathbf{c} \triangleq \mathbf{q}, \quad \dot{\mathbf{y}} \triangleq \dot{\mathbf{q}}, \quad (3.1)$$

where $\mathbf{q} \in \mathbb{R}^7$ is the arm configuration vector and $\dot{\mathbf{q}} \in \mathbb{R}^7$ is the arm's joint velocities vector.

A typical control objectives hierarchy for a generic manipulation action $\mathcal{A}_m$ is the following:

- $\mathbf{o}_1)$ $[\mathbf{r}, \leq, \mathbf{S}]$ Arm joints limits
- $\mathbf{o}_2)$ $[\mathbf{r}, \leq, \mathbf{S}]$ Arm obstacle avoidance
- $\mathbf{o}_3)$ $[\mathbf{r}, \leq, \mathbf{S}]$ Arm manipulability
- $\mathbf{o}_4)$ $[\mathbf{r}, =, \mathbf{A}]$ Tool frame position control
- $\mathbf{o}_5)$ $[\mathbf{r}, =, \mathbf{A}]$ Tool frame orientation control

where we have just task from two different categories: safety and action defining tasks. All tasks here are reactive ones.

Control Objective: Arm Joints Limits

The manipulator must operate within its joint limits, which means having the following inequality control objectives fulfilled

$$\begin{cases} q_i \geq q_{i,m} \\ q_i \leq q_{i,M} \end{cases} \quad i = 1, \dots, l, \quad (3.2)$$

where q_i is the i -th joint variable, $q_{i,m}$ is the lower bound and $q_{i,M}$ is the higher one for the joint i , and where l is the total number of joints of the manipulator.

To define the joints limits objective inside our task priority framework, standing the ‘ jl ’ subscript for joints limits objective, the associated reference rate and Jacobian matrix are:

$$\dot{\mathbf{x}}_{jl} = \gamma_{jl}(\mathbf{q}_{ref} - \mathbf{q}), \quad \mathbf{J}_{jl} = \mathbf{I}^{l \times l} \quad (3.3)$$

where $\mathbf{q}_{ref} = [q_1, \dots, q_l]$ is the array of joint reference values (which generally coincide with the centre of the joints motor interval), and γ_{jl} is the reference error gain. In this case, since the task is directly acting at the joint level, the Jacobian will simply be an $l \times l$ identity matrix. To avoid over-constraining the system when the arm is not near its joint limits we can define the following activation function for this task:

$$a_{jl,i} = \sigma_{\downarrow}(q_{i,m}, q_{i,m} + \Delta_{jl}) + \sigma_{\uparrow}(q_{i,M} - \Delta_{jl}, q_{i,M}), \quad (3.4)$$

where $a_{jl,i}$ the elements of the activation matrix $\mathbf{A}_{jl}$ as in (2.9), and we are assuming that the two sigmoid functions are not overlapping for points where the derivative is different from zero (i.e. the maximum value is still 1).

Control Objective: Manipulability

The arm must maintain a good dexterity, to be able to perform the manipulation tasks without incurring into the problems related to the kinematic singularities, which means to keep the manipulability measure μ [48] above a minimum threshold, thus fulfilling the following inequality control objective. In formulas:

$$\mu > \mu_{min}, \quad \text{with} \quad \mu = \sqrt{\det \{ \mathbf{J}(\mathbf{c}) \mathbf{J}(\mathbf{c})^T \}}, \quad (3.5)$$

where μ_{min} is the minimum threshold for the manipulability and $\mathbf{J}(\mathbf{c})$ is the arm Jacobian. Then, using ‘ m ’ as subscript for our manipulability control objective, the reference rate and Jacobian matrix [49] are defined as:

$$\dot{\mathbf{x}}_m = \gamma_m(\mu_{min} - \mu), \quad \mathbf{J}_{m,i} = \text{trace} \left\{ \frac{d\mathbf{J}}{dq_i} \mathbf{J}^\# \right\} \mu, \quad (3.6)$$

where $\mathbf{J}_{m,i}$ is the i -th element of the $l \times 1$ manipulability Jacobian, being this control objective a scalar one. In this case the element of the activation matrix will be the following:

$$a_{jl,i} = \sigma_{\downarrow}(\mu_{min}, \mu_{min} + \Delta_m), \quad (3.7)$$

Remark (Defective Manipulators). Let’s note that for defective manipulators (i.e. with less than 6 degrees of freedom), the manipulability measure as in (3.5) will

always be zero, since the result of the Jacobians multiplication will give a 6×6 matrix that will not have full rank. For this reason, for defective manipulators where the number of joints is $l < 6$, the manipulability measure μ_{def} will be the following:

$$\mu_{def} = \sqrt{\det\{J(\mathbf{c})^T J(\mathbf{c})\}}, \quad (3.8)$$

where the transpose operator makes so that the output is an $l \times l$ matrix with full rank.

This for example is the case for one of the robots used in the experiments of Chapter 4, the YouBot, where the arm has 5 degrees of freedom.

Another possibility to maintain a good dexterity is to define a preferred shape of the manipulator, where it is known that the arm is characterized by a good manipulability, and then try to maintain a suitable distance between the current arm configuration and the preferred one.

Control Objective: Arm Joint Obstacle Avoidance

A well known and critical control objective in robotics is obstacle avoidance. Using the control framework we can design a joint avoidance task using scalar control objective, which control the distance from each joint to the bounding box of the obstacle. For example we could use a spherical bounding box of radius r to delimit our obstacle. Now, for simplicity, let's examine the case for a single joint.

Being i the joint's index we want to control the avoidance of, and $\mathbf{n} = \mathbf{d}/\|\mathbf{d}\|$ the distance versor from the joint frame to the obstacle frame, the reference rate and Jacobian for this task can be defined as follows:

$$\dot{\hat{x}}_{ja} = \gamma_{ja}((r + \Delta_{ja}) - \|\mathbf{d}\|), \quad \mathbf{J}_{ja} = \mathbf{n}^T \mathbf{J}_i(\mathbf{c}) \quad (3.9)$$

where $\dot{\hat{x}}_{ja}$ is the task control reference for the obstacle avoidance task, γ_{ja} is the control gain, h is the obstacle height and Δ_{ja} is a constant displacement that defines a safety distance from the obstacle. Pre-multiplying the arm Jacobian by the distance vector transposed we are requesting to the system to generate a velocity which points away from the object bounding box.

The associated activation function assumes then the following form:

$$a_{ja,i} = \sigma_{\downarrow}(r, r + \Delta_{ja}) \quad (3.10)$$

where, for each a , a decreasing sigmoid shaped function activates the tasks when the joint approaches the Δ_{ja} -transition zone around the obstacle box. Using this

task we are then able to manage the arm obstacle avoidance for each joint, therefore addressing contribution C_2 .

Control Objective: End-effector position and orientation

Of course, the most important task for completing a mission will be to correctly position and orient the robot's end-effector. In this case the task definition will be quite straightforward. Having position and orientation two different unit types we will design two separate tasks. This is the reason why in the task list 3.1 they are listed separately, having the position as our main priority and successively try to achieve the correct orientation. In this case the tasks will be always active so we don't have to define the activation function. Having the following Cartesian velocity reference for our end-effector:

$$\dot{\mathbf{x}} = \begin{bmatrix} \omega_x & \omega_y & \omega_z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T, \quad (3.11)$$

the two tasks will very simply be:

- the end-effector position task (ep)

$$\dot{\mathbf{x}}_{ep} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T, \quad \mathbf{J}_{ep} = \mathbf{J}_{lin} \quad (3.12)$$

- the end-effector orientation task (eo)

$$\dot{\mathbf{x}}_{eo} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T, \quad \mathbf{J}_{eo} = \mathbf{J}_{ang} \quad (3.13)$$

where $\mathbf{J}_{lin}$ and $\mathbf{J}_{ang}$ are the linear and angular sub-matrices of the arm's Jacobian.

3.1.1 Experimental Results

Let's now have a look to some experimental results obtained by using the task hierarchy mentioned in the task hierarchy list of 3.1.

The robot is commanded to perform a diagonal movement on the horizontal plane, in order to reach the desired final position. Figure 3.1a shows that during such a movement, one of the joint's arm approaches its end of race, causing the activation of the corresponding row in the joint limits avoidance task. Figure 3.1b shows the joint's position and the bounds for the start of task activation (-0.41 rad) and its complete activation (-0.61 rad). The figure thus show how the task successfully avoids the joint limit. The generated joint velocities are reported in Fig.

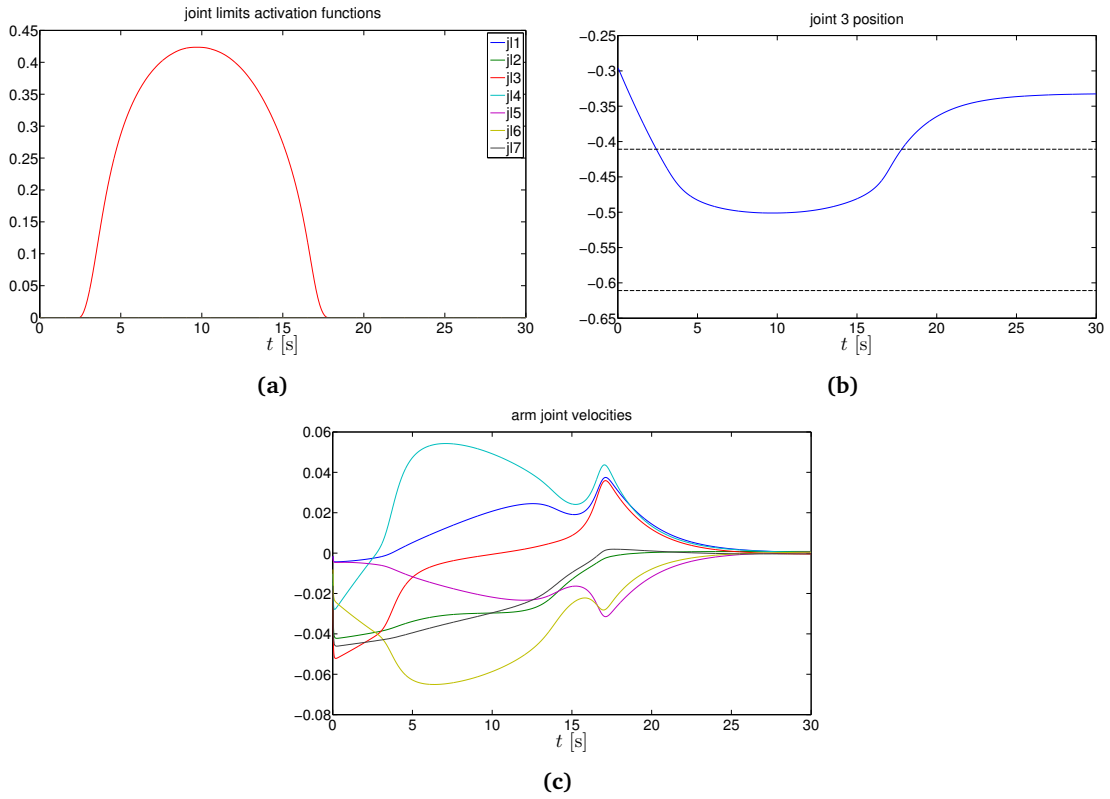


Figure 3.1: Single arm experimental trial: (a) activation functions for the joint limits task, (b) values of the joint near its physical limit (c) time history of the joint velocities during the trial.

3.1c, which are continuous and free of chattering phenomena even during the activation of the joint limit task. In this case there were no obstacles in the area so the results for that task have been neglected. A more interesting analysis for the avoidance task can be found in the next sections.

3.2 Dual-Arm Control

As outlined in the introduction, the framework is easily scalable to more complex systems than a single arm case. In this section in fact a dual arm control will be investigated and tested experimentally. In particular, we will see how it's possible, by designing an appropriate task, to use a bimanual robot for manipulating a rigid object. Of course now all the variables as configuration vectors, activation functions and Jacobians will have a 'double' size, having to fit all the left and right arm control variables.

As before, to describe the setup, in these bimanual experiments we can define the following quantities for the KCL:

- the system configuration vector $\mathbf{c} \in \mathbb{R}^n$ of the dual arm robot is

$$\mathbf{c} \triangleq \begin{bmatrix} \mathbf{q}_L \\ \mathbf{q}_R \end{bmatrix}, \quad (3.14)$$

where $\mathbf{q}_L \in \mathbb{R}^l$ is the left arm configuration vector and $\mathbf{q}_R \in \mathbb{R}^r$ is the right arm configuration vector.

- the system velocity vector $\dot{\mathbf{y}} \in \mathbb{R}^n$ of the robot is

$$\dot{\mathbf{y}} \triangleq \begin{bmatrix} \dot{\mathbf{q}}_L \\ \dot{\mathbf{q}}_R \end{bmatrix}, \quad (3.15)$$

where $\dot{\mathbf{q}}_L \in \mathbb{R}^l$ and $\dot{\mathbf{q}}_R \in \mathbb{R}^r$ are the left and right arm's joint velocities vectors. In this setup, having two identical 7 DOF arms, we will have $n = l + r = 14$.

Let's now proceed to the description of our task hierarchy list for this kind of scenario. For our *coordinated transportation* action $\mathcal{A}_{ct}$ we have defined the following prioritized list of tasks:

- $\mathbf{o}_1)$ [r, =, C] Rigid grasp constraint
- $\mathbf{o}_2)$ [r, $\leq$, S] Arm joints limits
- $\mathbf{o}_3)$ [r, $\leq$, S] Arm obstacle avoidance
- $\mathbf{o}_4)$ [r, $\leq$, S] Arm manipulability
- $\mathbf{o}_5)$ [r, =, A] Tool frame position control
- $\mathbf{o}_6)$ [r, =, A] Tool frame orientation control

where tasks are listed in order of decreasing priority and in which we have recalled the task classes we have defined. Defining the above *action*, the HLL is relieved from planning obstacle avoiding trajectories for the joints, which could also limit excessively the degrees of freedom of the arm, simplifying the implementation of the HLL itself.

As seen in Section 2.6, to define a control task, the three main quantities to be taken into account are: the control objective, the task Jacobian and the activation function. Let's see how these two quantities have been defined for the rigid grasp and obstacle avoidance tasks.

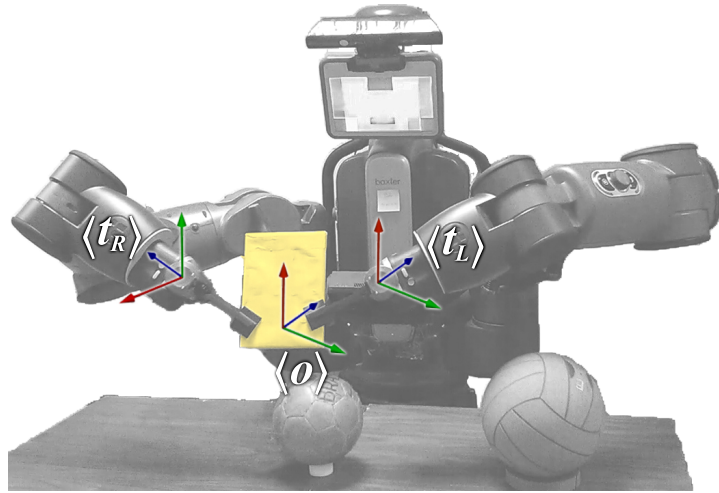


Figure 3.2: Frames involved in the rigid grasp manipulation: the left and right end-effector frames $\langle t_L \rangle$ and $\langle t_R \rangle$, and the object frame $\langle o \rangle$.

Control Objective: Bimanual Rigid Grasp

To perform a bimanual rigid object manipulation, once a rigid grasp has been performed, we can define a common object frame that will be used as a control reference frame for both the arms. In fact, to impose the rigid grasp constraint to be respected, what we can do is to fix the relative velocity of the two coinciding tool frames to be zero. In other words, we are exploiting the redundancy of the two arms to produce motions which are minimizing the misalignment between the two tool frames. Let us define $\langle t_L \rangle$ and $\langle t_R \rangle$ be the frames attached to left and right arm's tools and $\langle o \rangle$ a frame rigidly attached to the object, as shown in Figure 3.2. Having the generic tool to object transformation matrix ${}^t_o T$, we can define the *rigid body transformation matrix* $S_{t/o}$, as follows:

$$S_{t/o} = \begin{bmatrix} I^{3 \times 3} & \mathbf{0}^{3 \times 3} \\ [{}^t_o \mathbf{R} r_{t/o} \wedge] & I^{3 \times 3} \end{bmatrix}, \quad (3.16)$$

where the wedge symbol $\wedge$ represents the skew-symmetric matrix operator. This matrix simply moves the tool centre point (TCP) from the tool frame of the arm (whether $\langle t_L \rangle$ or $\langle t_R \rangle$), to the *common* object frame $\langle o \rangle$. We can then define two new arms Jacobians for the left and right arm, now acting on the object frame, as:

$$\tilde{J}_L = S_{L/o} J_L, \quad \tilde{J}_R = S_{R/o} J_R \quad (3.17)$$

In the condition of having the object firmly grasped by both the end effectors, the following differential constraints are imposed as consequence of the geometric

ones:

$$\dot{\mathbf{x}}_t = \tilde{\mathbf{J}}_L \dot{\mathbf{q}}_L = \tilde{\mathbf{J}}_R \dot{\mathbf{q}}_R, \quad (3.18)$$

with $\dot{\mathbf{x}}_t$ the object velocity with components on $\langle t \rangle$ and $\tilde{\mathbf{J}}_L, \tilde{\mathbf{J}}_R$ the arms Jacobians with respect to $\langle t \rangle$. We can then define:

$$\dot{\mathbf{x}}_{rg} = 0, \quad \mathbf{J}_{rg} = \begin{bmatrix} \tilde{\mathbf{J}}_L & -\tilde{\mathbf{J}}_R \end{bmatrix}, \quad (3.19)$$

where $\dot{\mathbf{x}}_{rg}$ is the task control reference for our rigid grasp and $\mathbf{J}_{rg}$ the task constraint Jacobian. What we are achieving, by defining this control reference and Jacobian, is to constantly tend to zero out the velocity difference between the left and right arm, in fact, by multiplying $\mathbf{J}_{rg}$ by our system velocity vector $\begin{bmatrix} \dot{\mathbf{q}}_L & \dot{\mathbf{q}}_R \end{bmatrix}^T$, what we get is the difference between the left and right arm velocities projected in the object frame, and we are imposing it to be zero. In this way we ensure that the kinematic grasp constraint is respected, therefore satisfying the functional requirement $\mathbf{C}_3$.

In this case, while the object is grasped, we want the objective to be always active so the activation function will be always equal to 1, and we may just use an action dependent activation as explained in section 2.5.1.

Control Objective: Dual Arm Obstacle Avoidance

The definition for the joint avoidance for the dual arm case derives directly for the general joint avoidance defined in the previous section. In this case of course we will have two references for each joint and a double sized Jacobian:

$$\begin{cases} \dot{\mathbf{x}}_{ja,L} = \gamma_{ja}((r + \Delta_{ja}) - \|\mathbf{d}_L\|) \\ \dot{\mathbf{x}}_{ja,R} = \gamma_{ja}((r + \Delta_{ja}) - \|\mathbf{d}_R\|) \end{cases} \quad \mathbf{J}_{ja} = \begin{bmatrix} \mathbf{n}_L^T \mathbf{J}_L & 0 \\ 0 & \mathbf{n}_R^T \mathbf{J}_R \end{bmatrix} \quad (3.20)$$

3.2.1 Experimental Results

The experimental trials carried out in these chapter have been part of a joint collaboration with another PhD student, which focuses on human-robot interaction research and AI techniques. The main challenge from the point of this work has been to develop the appropriate control interfaces to make the software development and integration as smooth as possible. The contribution proposed lays at the kinematic level, but for a better understanding I will briefly recall how this layer operates.

Two different set of experiments were performed using the Baxter, both of them involving bimanual coordinated object transportation. In the first set the task for the robot was to avoid the obstacle detected with a Kinect camera and safely move a bi-manually grasped object from a point to another. In the second set of experiments instead a human-robot interaction task was taken into account, where the robot had still to avoid obstacles but in the meanwhile holding a working piece in a specific position to let the human operator work on it.

For this experiment the KCL software architecture is running on a control PC which is connected through Ethernet to the robot and communicating using the ROS publish/subscribe mechanism. The implementation details will be described more in details in the dedicated Chapter .

Description of the HLL setup

The Higher Level Layer (HLL) we introduce here takes care of the robot's own perception, planning and actions so that it behaves as an individual agent. HLL is composed of three modules: Object and Scene Perception (OSP), Path Planning and Mission Control. This layer operates as follows.

The OSP receives the point cloud data from the Kinect sensor, and gives out the geometric description of the objects in the scene continuously, as shown in Figure 3.5f. The path planner receives the working space region, the final goal region, the obstacle regions described as cubes and the initial position of the path from the mission controller. It computes the path based on the RRT* method, and returns the resulting obstacle-free path points in space. The object, manipulated by the two robot's arms, should follow these path points. The mission control computes the relative frame of the object with respect to left and right end effectors frames that are necessary for coordinated control of the object (based on data provided by controller and OSP). The mission control passes the object path points (found by path planner), the aforementioned relative frames, and the obstacle regions (in working space) to the KCL. When the object reaches a way-point the the KCL informs the mission control and a new set pose is sent to it, and so forth, till the final pose is reached.

When the planner receives the objects specifications from the OSP it constructs bounding boxes around them, also adding a safety margin distance. The obstacles relevant parameters are also sent to the KCL so that it can build its own bounding boxes for joint obstacle avoidance. Moreover, for-bi the manual control of the robot

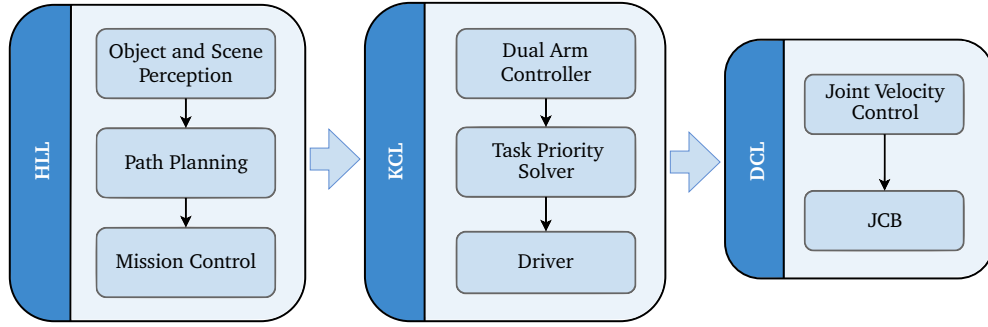


Figure 3.3: The three-layered architecture. In the HLL module the Scene Perception module sends the information on the current entities present in the scene to the Path Planning module which then communicates with the Mission Control module to interact with the kinematic layer. The KCL translates the commands received to control objectives and produces the reference joint velocities for the underlying dynamic layer. The DCL follows the given velocity at its best sending the final commands to the Joint Control Boards (JCB).

arms it is necessary to provide to the KCL the object frame position. To this goal we attached a frame to the centre of mass of the manipulating object that, being a plane, is consistent with its two principal dimensions.

Bimanual Experiments (I)

As shown in Figure 3.4 in this experiments we used the Baxter robot for manipulation, and a Kinect sensor mounted on the robot to get the point cloud. There are two balls in the scene playing the role of the obstacles and a plate as the object to manipulate. The robot arms should transport the plate coordinately using a bimanual control task; and while the object is avoiding the obstacles thanks to the HLL level, the elbow joints of the robot arms should avoid the obstacle reactively at the KCL level.

We have done two successive experiments to show how the HLL and KCL together solve the avoidance task through planning and in a reactive manner, respectively. In particular, in the first experiment the elbow obstacle avoidance task has not been included while it is in the second one.

Figures 3.5a and 3.5c show that, in both experiments, the object frame is avoiding the obstacles within the workspace and reaching its final goal by following the path generated by the path planner. Figure 3.5b shows the distance between the obstacle and the joint. The object is following its path, but the elbow is getting closer to an obstacle in the working space. Figures 3.5d and 3.5e depict the distance between the elbow and the obstacle and the activation function. When the elbow joint distance get close to the threshold of the obstacle bounding box the

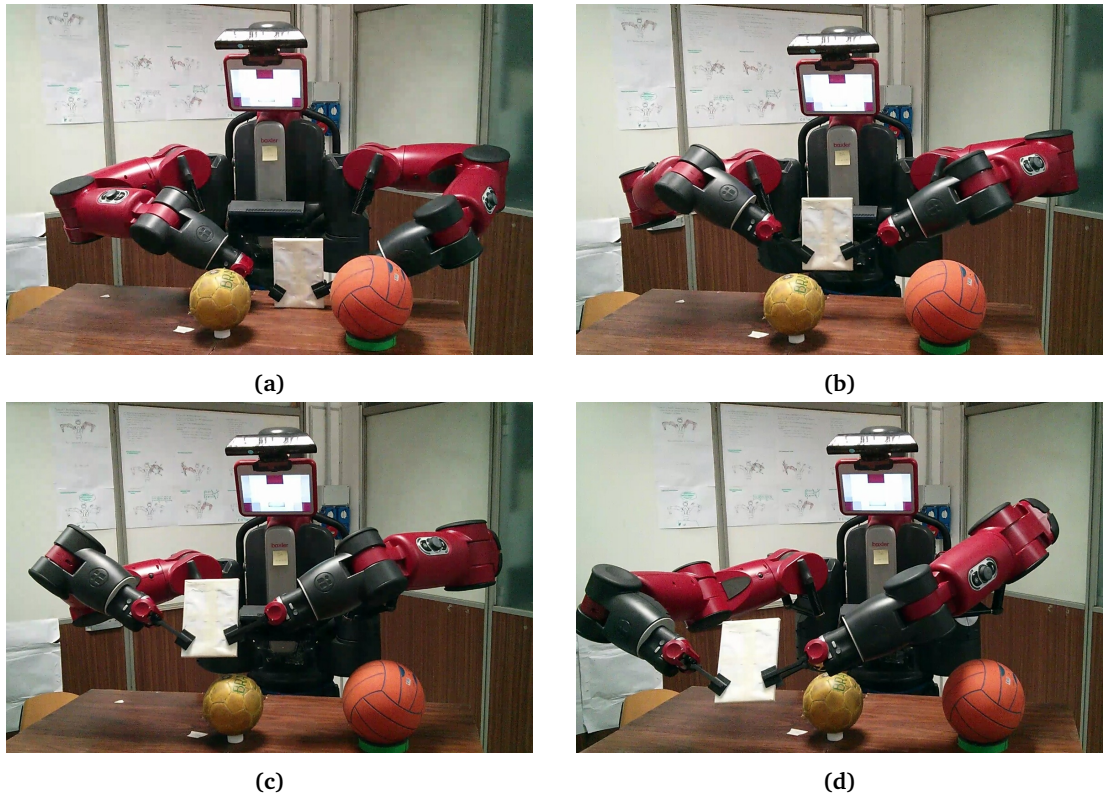


Figure 3.4: A series of successive still frames from one of the coordinated transportation experiments. As it can be seen in the last two frames 3.4c and 3.4d, the elbow is gradually raising its height to stay out of the bounding box defined for the obstacle.

task is activated, and while the object is following its path, the elbow is avoiding the obstacle within the defined threshold.

As the results of the experiments show, the KCL controls the robot arms to reactively avoid the elbow's obstacle, freeing the higher level from having to plan a path taking into account all the possible collisions.

Bimanual Experiments (II)

This second series of experiments is part of a bigger effort to provide a holistic and integrated solution to heterogeneous human robot collaboration challenges. The architecture developed takes the name of *FlexHRC*, based on a distributed hybrid reactive-deliberative architecture.

In this experiment, a human wearing a smart-watch and Baxter face each other on the opposite sides of a table, where bolts, wooden plates and screwdrivers are located in *a priori* known positions. The smart-watch is used to identify the human operator state based on the inertial data measured at the wrist (resting, screwing,

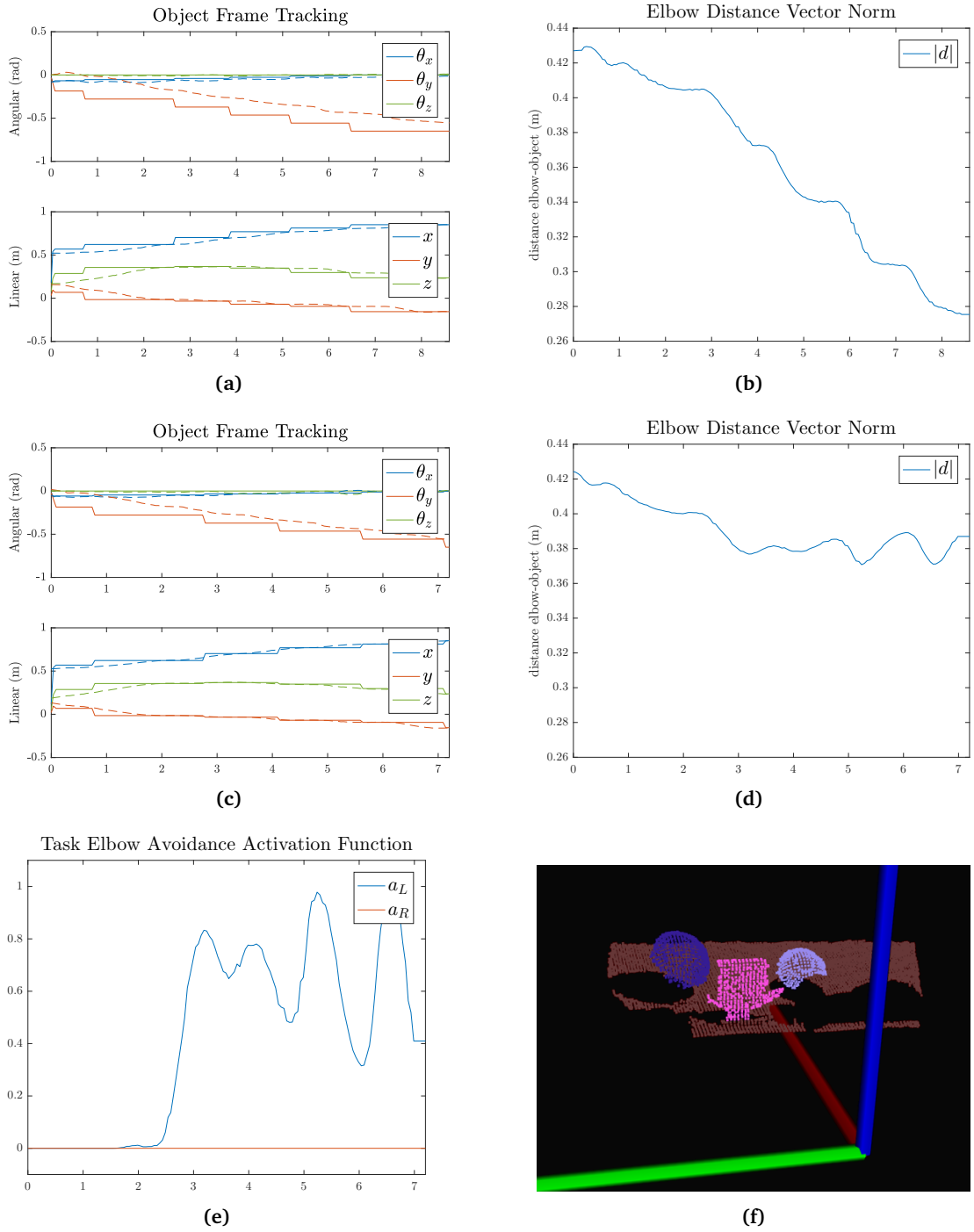


Figure 3.5: In figure 3.5f scene and objects recognized by OSP and relative frame of the vision system, each colour is identifying a different object. Figures 3.5a and 3.5b are for the experiments with obstacle avoidance only at the HLL level and figures 3.5c, 3.5d and 3.5e are for the experiments also with KCL reactive avoidance.

etc.). The goal of the task is to sink a bolt on a wooden plate using a screwdriver, and place the assembled piece in a final goal position on the table.

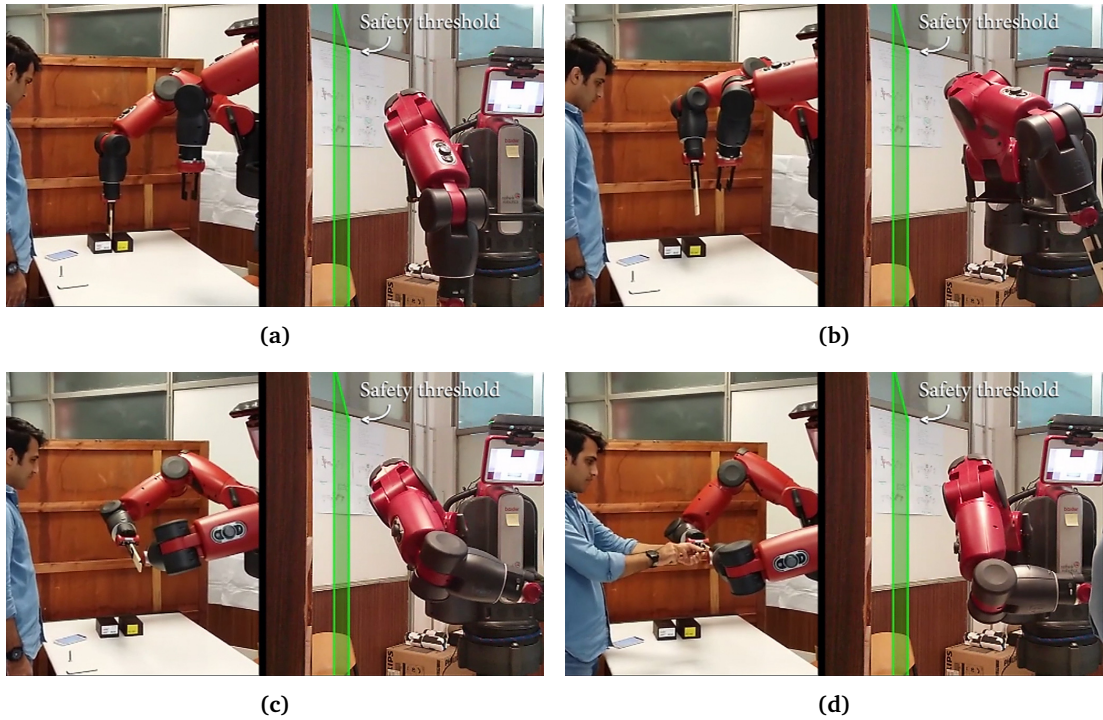


Figure 3.6: The HRI experiment where an obstacle is detected and avoided by the elbow joint while bi-manually holding a plate that's used by a human operator. The safety threshold has been highlighted in green. In figure 3.6d it can be observed how the right robot's forearm is higher than the left forearm which is avoiding the library.

The metric to test in this experiment is two-fold: on the one hand, assessing the KCL capabilities in dealing with situations requiring reactive control, e.g., obstacle avoidance, which does not need planning in the configuration space; on the other hand, doing so such not to jeopardize the overall cooperation context process, insofar as human-robot cooperation time is concerned. A few of the cooperation tasks have been repeated after the introduction of obstacles in the robot's workspace. An example is shown in Figure 3.6, where a lateral obstacle has been located on the right hand side of Baxter.

Figure 3.7 shows the trend of the activation function for the obstacle avoidance task, during both single-arm and dual-arm operations. It is possible to observe that the activation value never reaches the maximum, and the avoidance task is completed within the established thresholds. Therefore, the *Path Planning* can focus on the generation of Cartesian trajectories for the end-effectors or the object being manipulated, once grasped by both robot grippers, without planning in the configuration space.

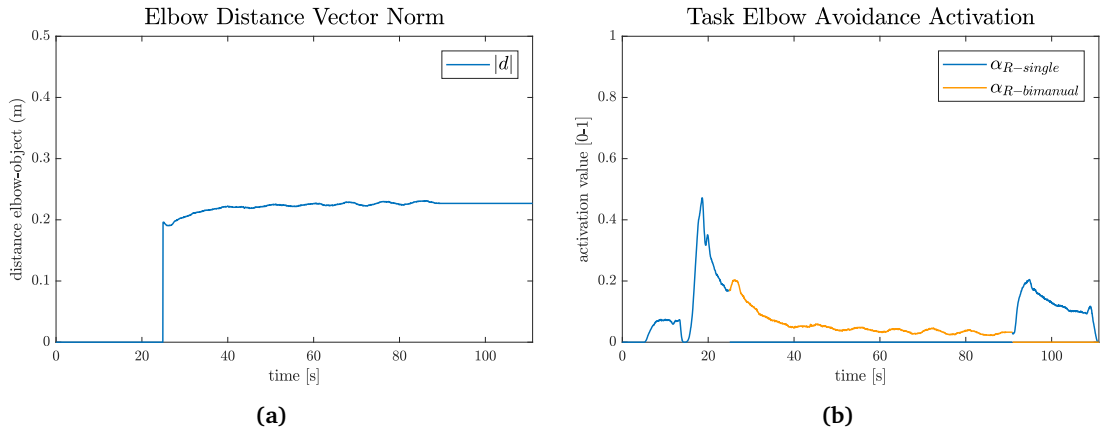


Figure 3.7: Elbow distance from the obstacle and activation function of the elbow avoidance task for the right arm: in blue for a single-arm operation, in orange for a dual-arm operation.

3.3 Mobile Manipulator Control

In this section we will see how the control framework easily scales to mobile robotic manipulators. In particular, taking into account contribution C_4 , this experimental simulation setup will also demonstrate how the proposed task priority framework, despite being at kinematic level, can manage interaction tasks. In particular the force control task will be placed in top of the action since we have to be sure that all the lower priority task, including the end-effector position control tasks, are congruent with the physical constraint of the surface.

Introducing now a floating vehicle, the system variables for this scenario are more similar to the general case. We can in fact describe the system as follows:

- The system configuration vector $\mathbf{c} \in \mathbb{R}^n$ as

$$\mathbf{c} \triangleq \begin{bmatrix} \boldsymbol{\eta} \\ \mathbf{q} \end{bmatrix}, \quad (3.21)$$

where $\boldsymbol{\eta} \in \mathbb{R}^m$ is the vehicle *generalized coordinate position vector* and $\mathbf{q} \in \mathbb{R}^l$ is the arm configuration vector. In this case having a 6 DOF vehicle plus a 7 DOF arm we will have $n = m + l = 13$.

- The system velocity vector $\dot{\mathbf{y}} \in \mathbb{R}^n$ as

$$\dot{\mathbf{y}} \triangleq \begin{bmatrix} \mathbf{v} \\ \dot{\mathbf{q}} \end{bmatrix}, \quad (3.22)$$

where $\mathbf{v} \in \mathbb{R}^m$ is the vehicle velocity vector and $\dot{\mathbf{q}} \in \mathbb{R}^l$ is the arm's joint velocities vector.

Following the aforementioned requirement of having interaction tasks at the top of the task hierarchy, the *kinematic force control* action $\mathcal{A}_f$ results to be:

- $\mathbf{o}_1)$ $[\mathbf{r}, =, \mathbf{C}]$ Constraint complying and Force regulation
- $\mathbf{o}_2)$ $[\mathbf{r}, \lesseqgtr, \mathbf{S}]$ Arm joints limits
- $\mathbf{o}_3)$ $[\mathbf{r}, \lesseqgtr, \mathbf{S}]$ Arm manipulability
- $\mathbf{o}_4)$ $[\mathbf{r}, \lesseqgtr, \mathbf{S}]$ Vehicle horizontal attitude
- $\mathbf{o}_5)$ $[\mathbf{r}, =, \mathbf{P}]$ Tool frame alignment with the object surface
- $\mathbf{o}_6)$ $[\mathbf{r}, =, \mathbf{A}]$ Tool frame position control
- $\mathbf{o}_7)$ $[\mathbf{r}, =, \mathbf{A}]$ Tool frame orientation control
- $\mathbf{o}_8)$ $[\mathbf{r}, \lesseqgtr, \mathbf{0}]$ Arm preferred shape

Let's now define the newly introduced tasks.

Control Objective: Force Regulation

Let us consider a surface, characterized by its normal $\mathbf{n} \in \mathbb{R}^3$ (in the point where the force is exerted), and let us call $\mathbf{J}_{el} \in \mathbb{R}^{3 \times n}$ the total linear Jacobian of the end-effector frame, comprehensive of both the vehicle and the manipulator contributions. For a moment, let us consider the end-effector to be bilaterally constrained on the surface, then the following holds

$$0 = \mathbf{n}^T \mathbf{J}_{el} \dot{\mathbf{y}}. \quad (3.23)$$

If the control vector $\dot{\mathbf{y}}$ is such that (3.23) would be violated, a force is exerted on the surface, and the actual $\dot{\mathbf{y}}$ will not converge to $\dot{\mathbf{y}}$ due to the constraint and consequently the tracking of the control task reference rates cannot be any more guaranteed.

For the aforementioned reasons, the control algorithm should take the constraint into account at the highest priority. Therefore, a *reactive* control task of the following kind can be placed at the top of the hierarchy:

$$S_1 \triangleq \left\{ \arg \text{R-min}_{\dot{\mathbf{y}}} \left\| (0 - \mathbf{n}^T \mathbf{J}_{el} \dot{\mathbf{y}}) \right\|^2 \right\}. \quad (3.24)$$

Introducing the above task will make the subsequent ones avoiding generating system velocities in the direction of the constraint, which would not be trackable by the DCL. In case of perfect tracking, no interaction with the environment will occur. Note that this is actually a reactive control task, since the control objective

corresponds to maintaining the geometric constraint which is guaranteed by the assumption made at the start of the section, hence the reference is always zero.

Let us now relax the assumption, and consider the actual *unilateral* constraint. In this case, a non-null interaction with the environment is desired to maintain contact with the surface through force regulation. To this aim, the idea is to impose a value $\dot{\mathbf{x}}_f$ rather than 0 as task reference of (3.24), obtaining

$$S_1 \triangleq \left\{ \arg \text{R-min}_{\dot{\mathbf{y}}} \left\| (\dot{\mathbf{x}}_f - \mathbf{n}^T \mathbf{J}_{el} \dot{\mathbf{y}}) \right\|^2 \right\}, \quad (3.25)$$

whose minimum norm solution is $(\mathbf{n}^T \mathbf{J}_{el})^\# \dot{\mathbf{x}}_f$. Its null space projector in fact, guarantees that all the lower priority task, including the end-effector position control tasks, are congruent with the physical constraint of the surface.

For the force regulation task, the following simple reactive reference rate has been used, and substituted in (3.25)

$$\dot{\mathbf{x}}_f \triangleq \gamma(\lambda^* - \lambda), \quad \gamma > 0, \quad (3.26)$$

where λ and λ^* the current and desired value of the force to be exerted on the surface, and γ a positive proportional gain. Note that, in this specific case, a proportional law is sufficient since the underlying dynamic control has been assumed already containing integrative actions.

Control Objective: Vehicle Horizontal Attitude

The vehicle attitude should be maintained within reasonable bounds, to avoid situations where the vehicle is upside-down. However, especially for vehicles and arms with comparable masses, such thresholds should be big enough to avoid excessive energy consumption to keep the vehicle perfectly horizontal whenever the arm moves its joints away from the centre of buoyancy and thus tilts the vehicle. The above outlined control objective requires the achievement of the following inequality

$$\|\boldsymbol{\varphi}\| \leq \varphi_M, \quad (3.27)$$

where $\boldsymbol{\varphi}$ represents the misalignment vector that the absolute world frame z axis forms with respect to the vehicle z axis one. Note that most of the ROVs (remotely operated vehicles) or AUVs (autonomous underwater vehicles) are passively stable in roll and pitch and these DOF are not controllable. For this reason, this objective should be considered only for fully actuated vehicles. In this case we will not expand on control reference and Jacobians since the Jacobian is directly the vehicle

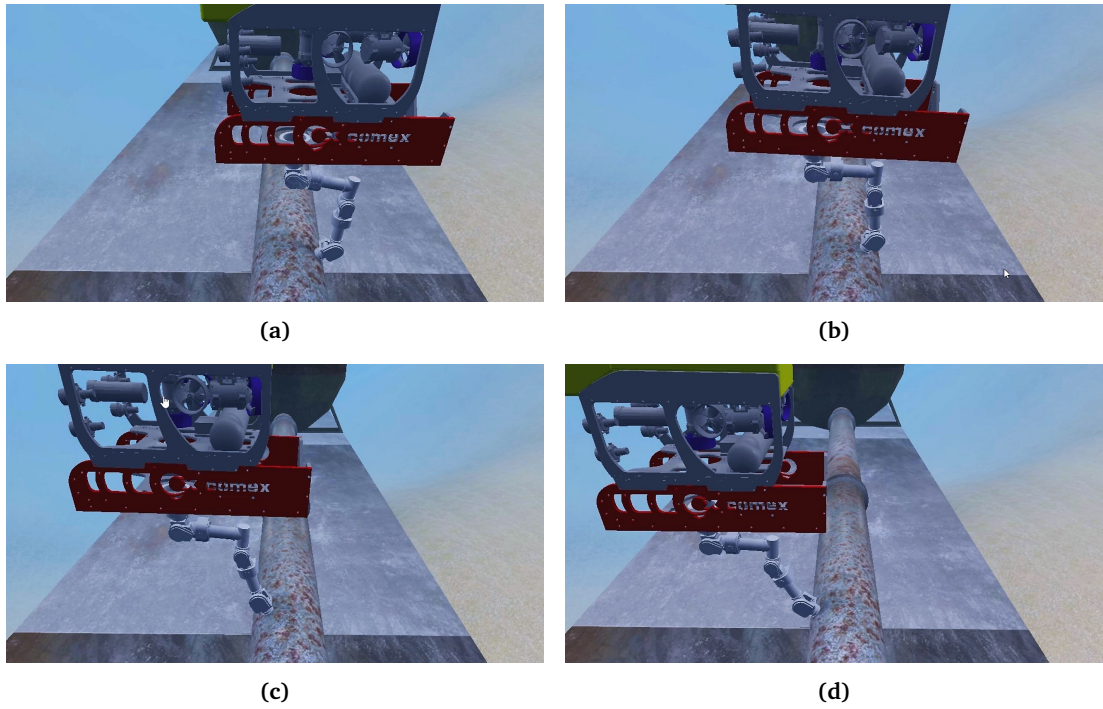


Figure 3.8: Screenshots of the UVMS as it performs the pipe welding inspection task.

Jacobian and the control vector is an error node having φ_M as reference, similarly to previous tasks. The same applies for the next control objective.

Control Objective: Vehicle Position Control

Another control objective is to have the vehicle frame $\langle v \rangle$ grossly aligned with a particular goal frame $\langle g_v \rangle$. This could be required in order to bring the vehicle close to the area where the manipulation activity needs to be carried out. This goal requires the achievement of the following inequality conditions

$$\|\mathbf{r}_v\| \leq r_{v,M} ; \|\boldsymbol{\vartheta}_v\| \leq \vartheta_{v,M}, \quad (3.28)$$

where $\mathbf{r}_v$ is the position error and $\boldsymbol{\vartheta}_v$ the orientation error.

3.3.1 Simulation Results

In this section we present some simulation results of the proposed kinematic control strategy. In fact, due to the unavailability of a physical wrist force/torque sensor, I've been exploiting an underwater simulative environment setup available in the lab. The reference mission is the inspection of a submersed pipeline using a UVMS. A reference path is defined without an a priori knowledge of the pipe, and the

expected results is that the end-effector follows the path projected on the pipe's surface. In all the simulations we have simulated a frictionless multi-point contact with the end-effector planar surface. All the simulated forces and moments are then transferred to a unique point on the end-effector's rigid body space, where we have assumed that a force/torque sensor has been placed.

The results presented in the following subsections have been obtained through dynamic simulations coded in MATLAB. In particular, the following settings have been used:

- Dynamic simulation and dynamic control loops were running at 1 kHz frequency.
- Dynamic simulation included linear and quadratic drag diagonal terms.
- Dynamic control was based on separate independent *proportional-integrative* loops, tuned around the nominal inertia of the links and vehicle.
- Each arm link inertia was modelled taking into account mass and reduction gears.
- Arm torques were saturated at $\begin{bmatrix} 45 & 125 & 50 & 50 & 10 & 10 & 4 \end{bmatrix}$ Nm (values are taken from the Graal Tech UMA commercial arm, see [50]).
- Kinematic control loop was running at 100 Hz frequency, based on the proposed task priority approach with velocity saturations.
- The considered velocity saturations were 1 rad/s for the arm joints, 0.2 m/s for the vehicle linear velocity and 0.2 rad/s for the vehicle angular velocities.
- Measurement noise is not included in the simulations.

Normally, to carry out a pipeline's weld inspection, an electromagnetic sensor is placed in contact with the weld, and made slide along all the surface. To do so with a robotised system requires that the UVMS regulates the contact force to a given value, to ensure contact without damaging the probe.

To carry out the inspection of the weld, a reference path has been defined in the horizontal plane, in correspondence of the projection of the weld. Therefore, there is no knowledge of the actual shape or radius of the pipe. The expected result is that the end-effector follows the path projected on the pipe's surface, adapting to the unknown surface. A frictionless *multi-point contact* between the end-effector

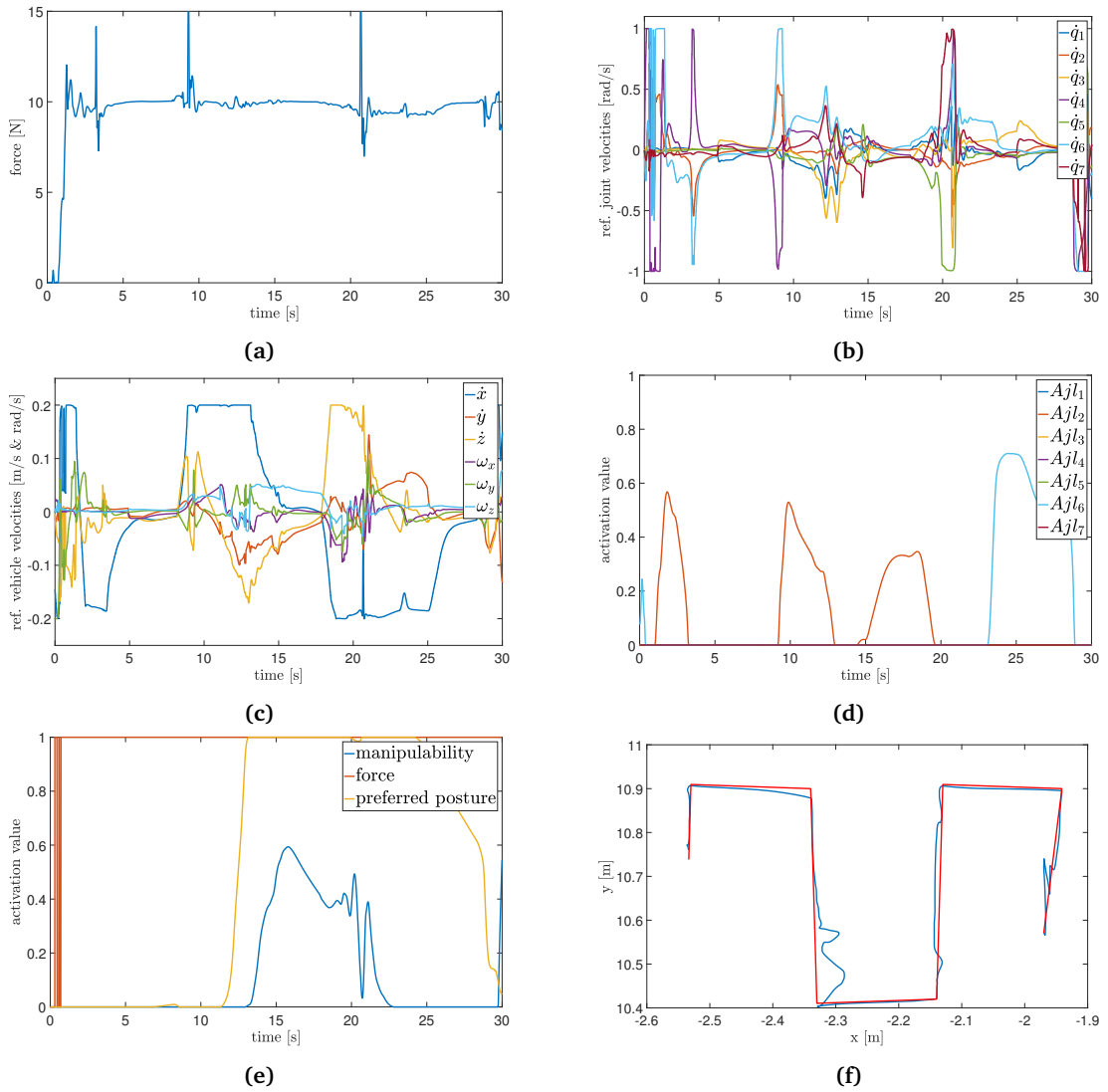


Figure 3.9: Simulation results: (a) the time behaviour of the norm of the force (reference value is 10 N), (b) the arm joint velocities (saturation value at 1 rad/s) (c) vehicle reference velocities (saturation value at 0.2 rad/s and 0.2 m/s) (d) the activation values of the joint limit tasks, (e) the activation values of the other tasks, (f) reference path (red) and actual trajectory (blue).

and the pipe surface was simulated. All the simulated forces and moments have been transferred to a unique point on the end-effector's rigid body space, where the presence of force/torque sensor has been assumed. Conversely to previous works [51], the dynamics of the thrusters have been simulated, based on the single state model [52] and taking the motor and propeller parameters from the Girona 500 [53] Seaeye thrusters SI-MCT01.

Screenshots of the UVMS performing the inspection task are reported in Figure

3.8, taken from the video available online¹.

The results are summarized in Figure 3.9, where in particular Figure 3.9a shows how the force is regulated close to the desired value of 10 N, while following the path on the pipe with a maximum deviation of approximately 3 cm, as shown in Figure 3.9f. The above results are achieved notwithstanding the activation of different safety tasks such as joint limits (see Figure 3.9d) and prerequisite ones such as manipulability (see Fig. 3.9e). Finally, the generated system velocity references take into account the saturations as it can be clearly appreciated in both Figure 3.9b and Figure 3.9c.

¹Video URL: <https://youtu.be/5hK2UMaon94>

EXTENSION TO MULTI-AGENT: THEORY

UNTIL now, we have always been considering a single agent involved in the completion of a particular task. In this chapter we will see how we can expand the theory introduced so far to control multiple agents acting cooperatively. Cooperative robots can be very useful for executing tasks that cannot be carried out by a single agent, as for example handling (i.e. transporting) big objects. Taking into account that each agent exploits the aforementioned single-agent control framework, the problem becomes that of finding a way to coordinate them for the execution of the mission. In the specific case of cooperative transportation, the problem is made even harder by the strict physical interaction occurring between the two robots via the common object, rigidly grasped.

In this section we shall tackle the extension of the above outlined control framework to the case of cooperating agents, when transporting and deploying a grasped, shared object. For the sake of simplicity, the discussion is limited to two mobile robots, even though it can be easily extended to more than two agents.

After the single agent control architecture has been presented in the previous section, let us now focus on its extension to the case of multiple cooperative mobile manipulators. For simplicity of discussion, the section will focus on two robots, as done in the experiments, but the approach can be scaled to multiple agents, so to enforce the C_5 contribution.

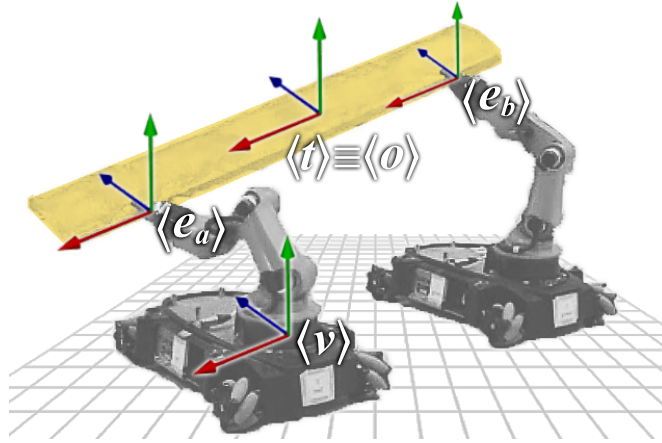


Figure 4.1: Frames involved in the cooperative mobile manipulation: the vehicle frame $\langle v \rangle$, the end-effector frames $\langle e \rangle$ and the tool control point $\langle t \rangle$ which coincides with object frame $\langle o \rangle$.

Assuming a firm object grasping by part of two agents a, b , the tool control points $\langle t_a \rangle, \langle t_b \rangle$ are assigned by the agents to coincide with the shared object fixed frame $\langle o \rangle$, that is $\langle t_a \rangle \equiv \langle t_b \rangle \equiv \langle o \rangle \triangleq \langle t \rangle$, as exemplified in Fig. 4.1.

In these conditions, the following differential constraints are imposed as consequence of the geometric ones

$$\dot{\mathbf{x}}_t = \mathbf{J}_{t,a} \dot{\mathbf{y}}_a = \mathbf{J}_{t,b} \dot{\mathbf{y}}_b, \quad (4.1)$$

with $\dot{\mathbf{x}}_t$ the object velocity with components on $\langle t \rangle$ and $\mathbf{J}_{t,a}, \mathbf{J}_{t,b}$ the system Jacobians with respect to $\langle t \rangle$.

Let us rewrite the second equation in (4.1) as

$$\begin{bmatrix} \mathbf{J}_{t,a} & -\mathbf{J}_{t,b} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{y}}_a \\ \dot{\mathbf{y}}_b \end{bmatrix} \triangleq \mathbf{G} \dot{\mathbf{y}}_{ab} = \mathbf{0}, \iff \dot{\mathbf{y}}_{ab} \in \ker(\mathbf{G}), \quad (4.2)$$

which represents the subspace where $\dot{\mathbf{y}}_{ab}$ is constrained to lay as a consequence of the firm grasp assumption. The definition of the constrained space $\mathbf{G}$ of the overall system velocities of both mobile manipulators clearly couples the solutions of the two subsystem, and it would be useful only in case of a centralized solution. However, such a possibility is discarded due to its required high rate of information exchange between the systems. Instead, the coordination policy proposed here is based on a reduced set of information that qualifies the evolution of the cooperation objectives, i.e. to comply with the grasp condition and to transfer the object to the desired location.

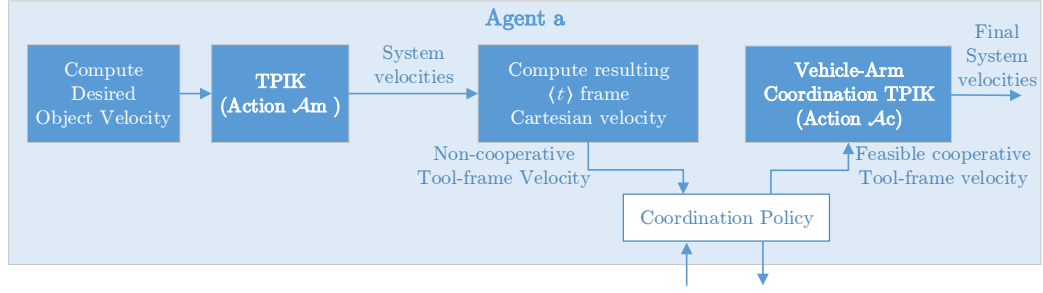


Figure 4.2: Block diagram of the cooperation, as seen from agent a . 1) a TPIK (action $\mathcal{A}_m$) is first solved, acting as if the agent was alone 2) the system velocity vector output of the TPIK is used to compute the *non-cooperative* tool-frame velocity 3) the non-cooperative tool frame velocity is processed according to the given *Coordination Policy*, which computes the *feasible cooperative* one through (4.7) and (4.9) 4) Agent a now solves a new TPIK (action $\mathcal{A}_c$) constrained to the feasible cooperative tool-frame velocity

Before proceeding with the coordination policy, let us consider the subspace of admissible Cartesian velocities of both systems

$$\dot{\mathbf{x}}_t = \mathbf{J}_{t,a} \mathbf{J}_{t,a}^\# \dot{\mathbf{x}}_t = \mathbf{J}_{t,b} \mathbf{J}_{t,b}^\# \dot{\mathbf{x}}_t, \quad (4.3)$$

which has been derived by (4.2) by eliminating the internal motion components within each subsystems, thus transferring the original constraint (4.1) from being expressed within the overall system velocity space, into the one expressed in the Cartesian space. Again, let us rewrite the second equation in (4.3) as:

$$\left(\mathbf{J}_{t,a} \mathbf{J}_{t,a}^\# - \mathbf{J}_{t,b} \mathbf{J}_{t,b}^\# \right) \dot{\mathbf{x}}_t \triangleq \mathbf{C} \dot{\mathbf{x}}_t = \mathbf{0} \quad (4.4)$$

thus implying $\dot{\mathbf{x}}_t \in \ker(\mathbf{C}) = \text{Span}(\mathbf{I} - \mathbf{C}^\# \mathbf{C})$, where the kernel of the matrix $\mathbf{C}$ (termed Cartesian constraint matrix) expresses the space of achievable object velocities at the current, overall-system constrained, configuration. Therefore, if a desired object velocity $\dot{\mathbf{x}}_t$ lying in this subspace is separately commanded to both agents *and* the corresponding non-reactive end-effector velocity tracking task is located at the *highest priority* in both agents task priority lists, then the object kinematic constraint is satisfied at kinematic level.

Let's now see proceed with the description of coordination algorithm phases.

4.1 Coordination Algorithm

For the development of the coordination policy the following assumptions are made:

- The coordination policy is a strategy shared and running simultaneously on all the robots, whose inputs are all the *non-cooperative* velocities, and whose output is the overall *feasible cooperative* velocity.
- All agents know the transformation matrix between the object goal frame $\langle g \rangle$ and the tool/object frame $\langle t \rangle$.
- All agents can evaluate the current reference feedback velocity $\dot{\mathbf{x}}_t$, which in case it were applied would asymptotically transfer the object to the desired goal location.

During each sampling intervals, the following sequential actions take place for the implementation of the proposed coordination policy, which is also sketched in Figure 4.2:

1. Each agent runs its own TPIK procedure (as detailed in Chapter 2), with its original task hierarchy $\mathcal{A}_m$ where safety and prerequisite tasks are located at higher priority than the end-effector motion.
2. Both agents compute the resulting, *non-cooperative*, tool-frame Cartesian velocities and transfer their computed quantities to the Coordinator.
3. The *feasible cooperative* velocity vector is computed according to the desired coordination policy.
4. Both agents run a new TPIK with a properly reorganized task hierarchy and actuate their resulting system velocity output.

In the next sections will take a closer look to the evolution of the above-mentioned coordination steps.

4.1.1 First Independent Optimization

In this first step the two mobile robots perform an independent optimization, as if each one of them were the only one acting on the object. In this case, they both solve a task hierarchy consisting of the individual objectives presented in Section 2.4, plus the vehicle distance objective, thus deliberately neglecting the kinematic constraint.

In this context, at each time instant, the outputs of the TPIK procedures for the two robots are the control vectors $\dot{\mathbf{y}}_a, \dot{\mathbf{y}}_b$. Both agents compute the resulting, *non-cooperative*, tool-frame Cartesian velocities that would be commanded to their respective tool-frames, if $\dot{\mathbf{y}}_a, \dot{\mathbf{y}}_b$ were applied:

$$\begin{aligned}\dot{\mathbf{x}}_{t,a} &= \mathbf{J}_{t,a} \dot{\mathbf{y}}_a, \\ \dot{\mathbf{x}}_{t,b} &= \mathbf{J}_{t,b} \dot{\mathbf{y}}_b.\end{aligned}\tag{4.5}$$

and their admissible velocities at the tool-frame $\mathbf{J}_{t,i} \mathbf{J}_{t,i}^\#$.

Note that, due to the different optimization conditions of the two systems, such velocities might not satisfy the kinematic constraint 4.1, i.e. $\dot{\mathbf{x}}_{t,a} \neq \dot{\mathbf{x}}_{t,b}$. Thus, in order to eliminate the drawback that could arise whenever $\dot{\mathbf{y}}_a, \dot{\mathbf{y}}_b$ were applied as system velocity references, possibly leading to unwanted object stresses, the execution of the second step of the procedure is proposed as hereafter described.

4.1.2 Coordination phase

The second step of the proposed coordination architecture requires the exchange of all the tool-frame velocities by all the agents, in this case $\dot{\mathbf{x}}_{t,a}, \dot{\mathbf{x}}_{t,b}$, and the subsequent computation of the final cooperative velocity using a shared coordination policy. Starting from these individual velocities we can in fact evaluate the common tool-frame velocity as

$$\dot{\hat{\mathbf{x}}}_t = \mathbf{f}(\dot{\mathbf{x}}_{t,a}, \dot{\mathbf{x}}_{t,b}),\tag{4.6}$$

where $\mathbf{f}$ is an a priori agreed fusion policy function. As an example, a convex combination can be employed.

The proposed policy is composed by the following sequence of steps:

1. Evaluating the *cooperative* tool-frame velocity vector

$$\dot{\hat{\mathbf{x}}}_t = \frac{1}{\mu_a + \mu_b} (\mu_a \dot{\mathbf{x}}_{t,a} + \mu_b \dot{\mathbf{x}}_{t,b}), \quad \mu_a, \mu_b > 0,\tag{4.7}$$

which corresponds to a weighted compromise between the two output velocities $\dot{\mathbf{x}}_{t,a}, \dot{\mathbf{x}}_{t,b}$. The rationale and the details underlying such a weighting policy will be given at the end of the description of this procedure.

If both agents have their safety objectives satisfied, and their current tool-frame Jacobians are non-defective, i.e. $\mathbf{J}_{t,i} \mathbf{J}_{t,i}^\# = \mathbf{I}$ for both $i = a, b$, then $\dot{\hat{\mathbf{x}}}_t = \dot{\mathbf{x}}_{t,a} = \dot{\mathbf{x}}_{t,b}$ independently from the weights, and the kinematic constraints are clearly satisfied.

Instead, in the general case, when $\dot{\mathbf{x}}_{t,a} \neq \dot{\mathbf{x}}_{t,b}$ their weighted sum (4.7) might not lay in the space of feasible object velocities, therefore it must be projected on such a subspace, as performed in the next two steps.

2. Evaluating the Cartesian constraint matrix

$$\mathbf{C} \triangleq (\mathbf{J}_{t,a} \mathbf{J}_{t,a}^\# - \mathbf{J}_{t,b} \mathbf{J}_{t,b}^\#) \quad (4.8)$$

3. Projection of the evaluated cooperative velocity vector $\dot{\mathbf{x}}_t$ on the feasible velocity space $\text{Span}(\mathbf{I} - \mathbf{C}^\# \mathbf{C})$ for the constrained object, thus obtaining

$$\dot{\mathbf{x}}_t = (\mathbf{I} - \mathbf{C}^\# \mathbf{C}) \dot{\mathbf{x}}_t \quad (4.9)$$

corresponding to the so-called *feasible cooperative* velocity vector $\dot{\mathbf{x}}_t$.

4. It transfers to both agents the computed velocity vector $\dot{\mathbf{x}}_t$.

The proposed choice for the assignment of weights μ_a, μ_b within 4.7 is the following one:

$$\begin{aligned} \mu_a &= \mu_0 + \|\dot{\mathbf{x}}_t - \dot{\mathbf{x}}_{t,a}\| \triangleq \mu_0 + \|\mathbf{e}_a\|, \\ \mu_b &= \mu_0 + \|\dot{\mathbf{x}}_t - \dot{\mathbf{x}}_{t,b}\| \triangleq \mu_0 + \|\mathbf{e}_b\|, \end{aligned} \quad (4.10)$$

with $\mu_0 > 0$ and where the norms $\|\mathbf{e}_i\|$ can be each one interpreted as a global measure of the difficulties that agent i has in tracking the original object feedback reference velocity, due to higher priority safety and prerequisite tasks. Thus, with the above weighting choice, the resulting feasible cooperative velocity $\dot{\mathbf{x}}_t$ will be closer to the one evaluated by the agent exhibiting the greatest difficulty in tracking the original desired object velocity, than to the one evaluated by the other agent.

For example, in case $\|\mathbf{e}_a\|/\|\mathbf{e}_b\| \rightarrow \infty$, then $\dot{\mathbf{x}}_t \rightarrow \dot{\mathbf{x}}_{t,a}$, which implies that agent a progressively imposes its evaluated tool-frame velocity as the cooperative one. Consequently, agent b will progressively follow what imposed by agent a at Cartesian level. Such a situation, and its opposite one, correspond to the extreme cases. In all the other situations, the weighting rule (4.10) provides an adequate compromise.

The above discussion is strongly based on the implicitly assumed property that nothing can change, for an agent, whenever an equality task is transferred to the highest priority, provided that its velocity is maintained equal to the result it provided when it was at a lower priority level. Such an assumed property, despite appearing as an evident one at a first glance, is proven at the end of this chapter, in Section 4.1.4.

4.1.3 Second Independent Optimization

After the exchange of their optimal independent tool-frame velocities and the computation of the common one $\dot{\mathbf{x}}_t$, both agents run a new TPIK, implementing the vehicle arm coordination of section 2.8. This time though, the original task-priority hierarchy has a non-reactive tool-frame velocity tracking at the *highest priority*. For example they could execute a coordinated manipulation action $\mathcal{A}_c$ so composed:

- $\mathbf{o}_1)$ $[\text{nr}, =, \mathbb{C}]$ Constrained tool-frame velocity
- $\mathbf{o}_2)$ $[\text{r}, \lesseqgtr, \mathbb{S}]$ Arm joints limits
- $\mathbf{o}_3)$ $[\text{r}, \lesseqgtr, \mathbb{S}]$ Arm obstacle avoidance
- $\mathbf{o}_4)$ $[\text{r}, \lesseqgtr, \mathbb{S}]$ Arm manipulability
- $\mathbf{o}_5)$ $[\text{r}, \lesseqgtr, 0]$ Arm preferred shape

Note that now both systems, independently, optimize their safety tasks constrained to having the Cartesian velocity $\dot{\mathbf{x}}_t$ at their end-effectors.

This approach allows to also subsequently further optimize the internal motions of the system for better achieving all the other tasks, for a given $\dot{\mathbf{x}}_t$. Following this idea, the two agents thus obtain a new couple of system velocity vectors $\dot{\mathbf{y}}_a$ and $\dot{\mathbf{y}}_b$ which guarantee

$$\begin{aligned}\dot{\mathbf{x}}_t &= \mathbf{J}_{t,a} \dot{\mathbf{y}}_a, \\ \dot{\mathbf{x}}_t &= \mathbf{J}_{t,b} \dot{\mathbf{y}}_b,\end{aligned}\tag{4.11}$$

and thus satisfy the object kinematic constraint 4.1.

Remark (Constrained tool-frame velocity). Since the end-effector task is at the top of the hierarchy, and the tracking of its velocity reference is guaranteed for the already stated reasons, one could ask why we have not used directly the original object reference rate $\dot{\mathbf{x}}_t$. The difference lies in the fact that $\dot{\mathbf{x}}_t$ only depends on the current tool frame $\langle t \rangle$ and goal frame $\langle g \rangle$ positions, while $\dot{\mathbf{x}}_t$ depends, indirectly, also on all the task references of both agents because it is the fusion 4.6 of the best Cartesian tool frame velocities 4.5 that each system can independently obtain considering all their other higher priority tasks.

4.1.4 Condition for Priority Inversion Invariance

Regarding the development of the coordination policy, let us make a remark on a particular change of task hierarchy. Let us consider an action $\mathcal{A} = \tau_A \prec \dots \prec \tau_K \prec$

$\dots \prec \tau_Z$. The TPIK procedure proposed in 2.6 is used to solve such an hierarchy of tasks, and let us call its output ${}^1\dot{\mathbf{y}}^*$, which is the optimal control vector satisfying the priority requirements. To such an optimal control vector the associated optimal task velocities are ${}^1\dot{\mathbf{x}}_i^* = \mathbf{J}_i {}^1\dot{\mathbf{y}}^*$, $i = A, \dots, Z$.

Let us also consider the following decomposition of ${}^1\dot{\mathbf{y}}^*$:

$$\begin{aligned} {}^1\dot{\mathbf{y}}^* &= \mathbf{J}_K^\# \mathbf{J}_K {}^1\dot{\mathbf{y}}^* + (\mathbf{I} - \mathbf{J}_K^\# \mathbf{J}_K) {}^1\dot{\mathbf{y}}^* \\ &= \mathbf{J}_K^\# {}^1\dot{\mathbf{x}}_K^* + (\mathbf{I} - \mathbf{J}_K^\# \mathbf{J}_K) {}^1\dot{\mathbf{y}}^*, \end{aligned} \quad (4.12)$$

which can be interpreted as follows: not only ${}^1\dot{\mathbf{y}}^*$ is the optimal control vector of the whole hierarchy, but it can also be seen as the optimal solution in the null space of task τ_K , constrained to have a task velocity ${}^1\dot{\mathbf{x}}_K^*$.

Now, consider a new task hierarchy where τ_K is at the highest priority, therefore $\tau_K \prec \tau_A \prec \dots \prec \tau_J \prec \tau_L \dots \prec \tau_Z$. Let us further impose that the reference for the task τ_K is chosen as ${}^1\dot{\mathbf{x}}_K^*$, i.e. the best task velocity that could be obtained when τ_K was at its original priority order. The manifold of solutions, after solving the first task τ_K in the new hierarchy is

$${}^2\dot{\mathbf{y}}_1 = \mathbf{J}_K^\# {}^1\dot{\mathbf{x}}_K^* + (\mathbf{I} - \mathbf{J}_K^\# \mathbf{J}_K) \dot{\mathbf{z}}, \quad \forall \dot{\mathbf{z}}, \quad (4.13)$$

and let us immediately notice that the task τ_K velocity is still ${}^1\dot{\mathbf{x}}_K^*$:

$${}^2\dot{\mathbf{x}}_K = \mathbf{J}_K {}^2\dot{\mathbf{y}}_1 = \mathbf{J}_K \mathbf{J}_K^\# {}^1\dot{\mathbf{x}}_K^* = {}^1\dot{\mathbf{x}}_K^*, \quad (4.14)$$

since ${}^1\dot{\mathbf{x}}_K^* \in \text{Span}(\mathbf{J}_K)$ and $\mathbf{J}_K (\mathbf{I} - \mathbf{J}_K^\# \mathbf{J}_K) = \mathbf{0}$. Therefore, (4.13) is the manifold of solutions of task τ_K constrained to have a velocity ${}^1\dot{\mathbf{x}}_K^*$, i.e. the same as (4.12).

At the end of the procedure, the optimal velocity vector will be of the form:

$${}^2\dot{\mathbf{y}}^* = \mathbf{J}_K^\# {}^1\dot{\mathbf{x}}_K^* + (\mathbf{I} - \mathbf{J}_K^\# \mathbf{J}_K) \dot{\mathbf{z}}^*, \quad (4.15)$$

where the value $\dot{\mathbf{z}}^*$ has been obtained for the optimization of subsequent tasks $\tau_A \prec \dots \prec \tau_J \prec \tau_L \dots \prec \tau_Z$. Since the relative priority order of all the other tasks is unchanged, the optimization on $\dot{\mathbf{z}}$ must yield $\dot{\mathbf{z}}^* = {}^1\dot{\mathbf{y}}^*$ and (4.15) will be the same as (4.12). Therefore, ${}^2\dot{\mathbf{y}}^* = {}^1\dot{\mathbf{y}}^*$ and the optimal task velocities will be exactly the same as in the original TPIK procedure. If, absurdly, this was not the case, then it would mean that even if the task velocity ${}^2\dot{\mathbf{x}}_K^* = {}^1\dot{\mathbf{x}}_K^*$, and the optimization structure is the same in (4.12) and (4.13), i.e. in the null space of task τ_K , there are two different sets of optimal task velocities for tasks $i = A, \dots, Z, i \neq K$.

MULTI-AGENT CONTROL: APPLICATIONS

IN this chapter we will see two applications for the multi-agent control case. The first one in Section 5.1 will deal with cooperative transportation task in an underwater simulation environment, using two free-floating UVMSs. In this case the simulation is intended to be a pre-test phase so the control is not running in real-time as it is developed in Matlab code. In the second section instead, moving more towards an industry/logistic scenario, the control framework is tested on a pair of YouBots by Kuka, using the on-board linux PCs with the control framework developed in C++ on top of the Ortos middleware as introduced in Chapter 1.

5.1 Floating Base Cooperative Transportation

In this section will see now how the framework can be applied to a scenario where two mobile manipulators carry out a cooperative transportation. For this first experimental trial it has been used a simulative environment. To demonstrate how the framework easily scales to more complex setups we will now add more tasks to our priority list and consider a series of different action being performed successively. Exploiting some of the utilities available in the laboratory due to the involvement in a National project on marine robotics, this tests have been carried out using an underwater simulator with floating base UVMSs cooperating in the transportation of a rigid pipe (the target object). For each individual agent the model is the same as describe for the single floating manipulator in section 3.3. Let's then directly

proceed to the definition of the actions.

The manipulation and transportation reference mission carried out by the team of UVMSs can be decomposed into the following actions:

1. Navigation ($\mathcal{A}_n$): the vehicle(s) should (concurrently) get in close proximity with the target object to be manipulated;
2. Grasping ($\mathcal{A}_g$): the UVMS(s) must (concurrently) perform the grasping of the (shared) object;
3. Transportation ($\mathcal{A}_t$): the UVMS(s) must transport the (shared) object to the target area;
4. Deployment ($\mathcal{A}_d$): whenever in close proximity with the target area, the (shared) object must be deployed in the required position.

At the end of the above phases, the UVMSs can leave the target area and each of them can be assigned to a new mission. For all the actions, the considered unified prioritized list, as seen in 2.5.1, of tasks for individually operating UVMS is the following one:

- $\mathbf{o}_1$) $[\mathbf{r}, \leq, \mathbf{S}]$ Arm joints limits
- $\mathbf{o}_2$) $[\mathbf{r}, \leq, \mathbf{S}]$ Vehicle minimum distance
- $\mathbf{o}_3$) $[\mathbf{r}, \leq, \mathbf{S}]$ Arm manipulability
- $\mathbf{o}_4$) $[\mathbf{r}, \leq, \mathbf{S}]$ Camera object centring
- $\mathbf{o}_5$) $[\mathbf{r}, \leq, \mathbf{S}]$ Vehicle horizontal attitude
- $\mathbf{o}_6$) $[\mathbf{r}, =, \mathbf{S}]$ Vehicle position control
- $\mathbf{o}_7$) $[\mathbf{r}, =, \mathbf{A}]$ Tool frame position control
- $\mathbf{o}_8$) $[\mathbf{r}, =, \mathbf{A}]$ Tool frame orientation control
- $\mathbf{o}_9$) $[\mathbf{r}, \leq, \mathbf{S}]$ Minimization of arm and vehicle movements

We must remark how the vehicle and end-effector position control, which are those most relevant for the execution of different mission phases are necessarily at a low priority. This is due to the fact that the higher priority tasks are all related to the safety of the system (joint limits, vehicle distances, horizontal attitude) or are needed to enable the system to actually execute the mission (manipulability, camera object centring). For the second independent optimization of the cooperative

algorithm, the same priority list is used, with the addition of end-effector task at the top of the hierarchy.

The single actions select their objectives as in table 5.1 using the action transition paradigm explained. The table represents which task are activated depending on the current phase of the system, i.e. the value of their corresponding $a_{(i)}^p$. We can see that many action are in common. Of course, for tasks that correspond to inequality control objectives, this does not immediately means that their corresponding activation function $a_{(i)}$ is equal to one, since it still depends on the actual value of their corresponding $a_{(i)}^i$. Indeed for such tasks we have that $a_{(i)} = a_{(i)}^i$. As an example, the camera centring *can* be active during the grasping phase, but if the error vector is within the pre-established thresholds, the task can have its activation value equal to zero.

Task	Action				
	Navigate	Grasp	Transport	Deploy	Step-back
Joint Limits	1	1	1	1	1
Vehicle Distance	1	1	1	1	1
Manipulability	1	1	1	1	1
Camera Centring	0	1	0	0	0
Horizontal Attitude	1	1	1	1	1
Vehicle Position	1	0	0	0	0
Tool frame Pose	0	1	1	1	1
Minim. Arm Velocity	1	0	1	0	0
Minim. Vehicle Velocity	0	1	0	1	1

Table 5.1: Action-dependant activation values

Let's now briefly explain some of the new introduced objectives.

Control Objective: Camera Occlusion

A further inequality objective that might be required for the manipulator is to keep its links away from the camera system's cone of vision (assumed fixed to the vehicle), to avoid unnecessary occlusions of the target object frame $\langle o \rangle$. This task depends on the specific configuration of the arm: unavoidable occlusions might occur when the end-effector is in close proximity to the target, but in other cases other types of avoidable occlusions may happen, for instance when the elbow of

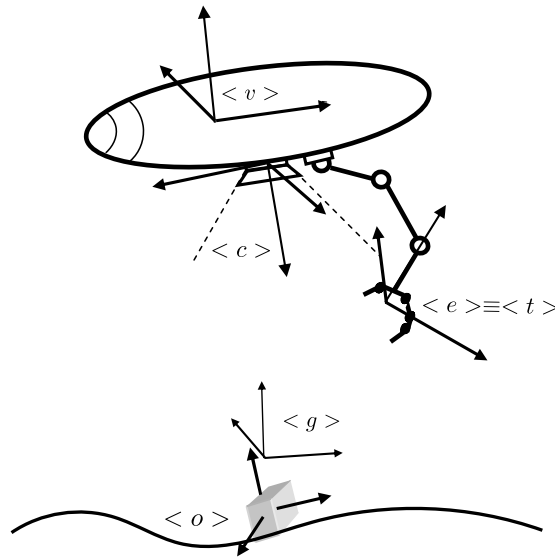


Figure 5.1: The UVMS and its relevant frames

the arm interferes with the camera well before the final grasping phases. In this case, a further inequality control objective can be introduced to force the elbow to stay outside the vision system cone.

Control Objective: Camera Centring

Since in the underwater environment the availability of an absolute positioning system is not always guaranteed, the navigation toward a particular target and especially the grasping of an object is often carried out by means of the stereo vision of the target itself. Thus, it becomes important that the visual contact between the system and the target is maintained as the vehicle approaches it. This can be translated by requiring that the target is grossly centred within the vision system. Toward that end, the control must ensure that the norm of the misalignment vector ξ between the vector joining the object $\langle o \rangle$ and the camera frame $\langle c \rangle$ with the z axis of the camera frame itself (supposed going outwards the camera image plane, as in Fig. 5.1) is within a maximum threshold, i.e.

$$\|\xi\| \leq \xi_M. \quad (5.1)$$

5.1.1 Simulation Results

In this section we present some results obtained in a simulation environment. In the scene we have a pipe object, whose reference frame $\langle o \rangle$ is placed at its centre and

is located in $\begin{bmatrix} 0.5 & 0.5 & 3 \end{bmatrix}$ w.r.t. a world inertial frame with the NED (north east down) coordinates (thus z represents the depth and is positive going downwards). The two UVMSs are required to grasp the pipe at a distance of 1.5 m from its centre, along the longitudinal axis.

In this section is presented a simplified simulation, where the UVMS is assumed kinematic. This allows to simulate a whole mission, where all the phases (navigation, grasping, transportation and deployment) are present. Furthermore, the algorithm is validated considering different communication parameters, including half-duplex communication, limited bandwidth and latencies.

In the simulation, in order to focus our attention on the execution of the whole mission, we make the following assumptions:

- the UVMSs are assumed kinematic;
- the UVMSs know their position and the goal position for the object at each sampling time;
- the UVMSs know their relative position (for the distance task);
- the communication channel is assumed error free and able to deliver the exchanged velocities at each control interval.

The simulated mission requires that, after the successful grasp of the object, the two UVMSs should bring the pipe in a pre-deployment position, located at $\begin{bmatrix} 3 & 5 & 2 \end{bmatrix}$ with a rotation along the z axis of the world frame of $\pi/2$, before deploying the object at the sea floor depth of 3 m.

The difference between the various actions is which of these tasks is active or not. The joint limits, minimum vehicle distance, manipulability and horizontal attitude tasks are always active; the camera centring task is only active during the grasping phase, while the vehicle position control is active only during the initial navigation phase. The end-effector position control is activated since the grasping action, and remains active for the rest of the mission. Finally, the tasks to minimize the arm or vehicle movements are alternatively activated: the arm movements are minimized during the navigation and transportation actions, while the vehicles movement are minimized during the grasping and deployment ones. Table 5.1 summarizes the action-dependant part of the activation values. The transitions $0 \rightarrow 1$ and vice versa are implemented with sigmoid functions as explained in 2.5.1,

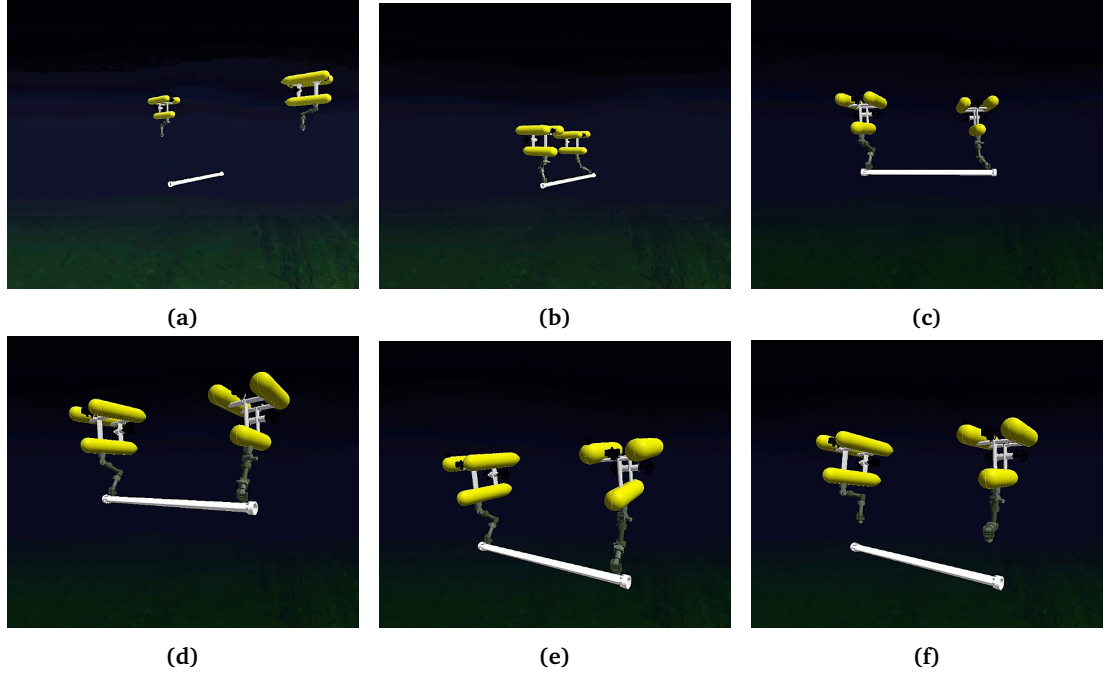


Figure 5.2: Screenshots of the simulation environment as the transportation mission is executed: (a) initial position $t = 0$ (b) successful grasp of the objects by both UVMS $t = 13.71$ (c) transportation toward the goal position $t = 20$ (d) pre-deployment position reached $t = 39.3$ (e) deployment of the pipe $t = 45$ (f) deployment completed $t = 55$.

parametrized with the action's time, i.e. they complete their transition within a couple of seconds since the start of their respective phase.

In addition to the actions presented we have added a small transition phase between the grasping and transportation actions, to introduce the switch between single UVMS control and the cooperative control policy. In particular, during this phase, the end-effector position control of the second independent optimization is activated at the top of the hierarchy, and smoothly deactivated at its original position. A similar, inverse, transition has been implemented at the end of the deployment of the object. The simulation also consider a final phase, after the object has been correctly placed on the sea floor, where the two UVMSs' end-effectors rise up from the object of around 0.5 m, now again acting independently from each other.

Figure 5.2 shows some screenshots of the simulation environment, showing all the main actions carried out in the mission. Figure 5.3 presents the generated system velocity references for both systems. It is important to remark how the generated velocities are free from discontinuities during each phase and especially from chattering phenomena, considering that the simulation implements a discrete

control. The only discontinuities that can be seen are those in correspondence of the activation and deactivation of the firm grasp constraint, which is an intrinsically discontinuous event.

In the figures we have highlighted the times at which there is a change of a mission phase: $t = 5.95$ is the end of navigation action and the start of the grasping action, which ends at $t = 13.2$. Then a small transition window allows the two UVMSs to start the coordinated control ($t = 15.7$). The two UVMSs reach a pre-deployment position at $t = 39.3$, and complete the deployment of the pipe on the sea floor at $t = 45.04$. A small transition window is again present to allow the UVMSs to switch from the coordinated control policy to the single free floating control ($t = 47.5$). Finally the two UVMSs move their respective end-effectors away from the pipe and the simulation is ended at $t = 55$.

The successive Fig. 5.4 depicts the time history of the activation values of each task for both systems. In particular we note how many different task are activated and deactivated throughout the mission execution:

- the minimization of the sub-system velocities of the arm and the vehicle are alternatively activated and deactivated depending on the action. This leads to the graphs of Fig. 5.3 where the arm is mostly used during the grasping and deploying tasks, while the vehicle is mostly used for accomplishing the transportation task;
- the camera centring task, which is an inequality control objective, is activated only during the grasping phase, however its value depends on the actual misalignment error and thus the activation value is not strictly one.

In the second simulation, the same virtual scene is used, but this time a sinusoidal velocity disturbance is added to the vehicles' velocities. The disturbance is directed as the inertial frame x axis, simulating an ocean time varying current. In this simulation, the disturbance compensation approach of Section 2.8 is implemented. This means that the arm is using the real vehicle velocity (assumed measurable) as a parameter for the computation of its parametrized control law. Figure 5.5 shows the obtained system velocities in this case. It is noteworthy to highlight how in this particular case, during the transportation phase $22.99 < t < 47.85$ the arm velocity is not zero as in Fig. 5.3, due to the fact that the arm is actively compensating the mismatches between the required vehicle velocity and the actual one. Finally, Fig. 5.6 reports the time history of the activation values of both UVMSs.

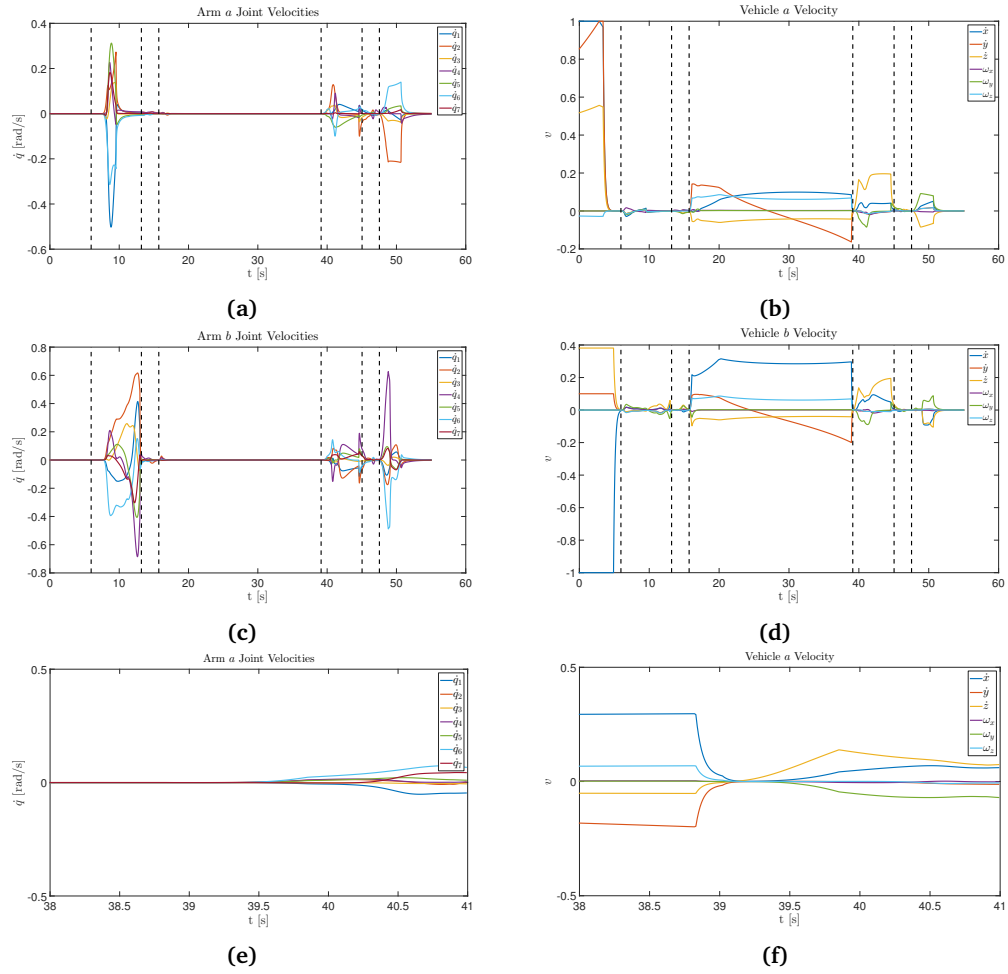


Figure 5.3: The system velocities during the first simulation trial, showing that the system does not suffer from discontinuities or chattering phenomena, except at the activation or deactivation of the firm grasp constraint. The vertical dashed lines separate the different mission phases. Time history of (a) arm a joint velocities $\dot{q}$ (b) vehicle b velocities $\mathbf{v}$, (c) arm b joint velocities $\dot{q}$ (d) vehicle b velocities $\mathbf{v}$ (e) a zoom of the arm b joint velocities and (f) of the vehicle b velocities in the interval $38 < t < 41$ to show the continuity of the approach.

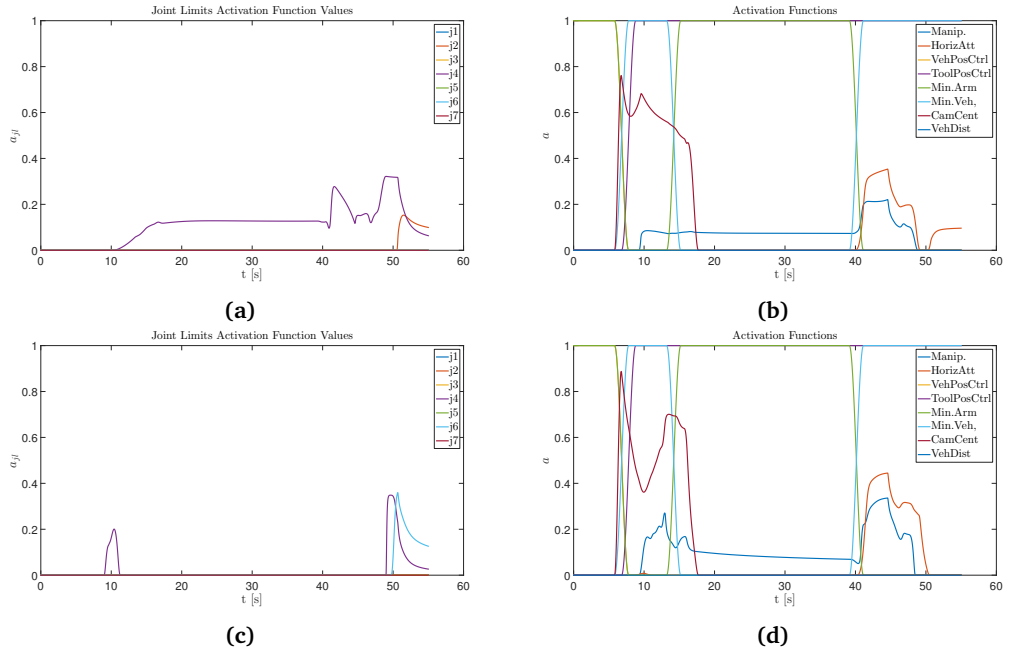


Figure 5.4: First trial. Time history of (a) arm a joint limits activation values and (b) the activation values of all the other tasks for UVMS a (c) arm b joint limits activation values and (d) the activation values of all the other tasks for UVMS b .

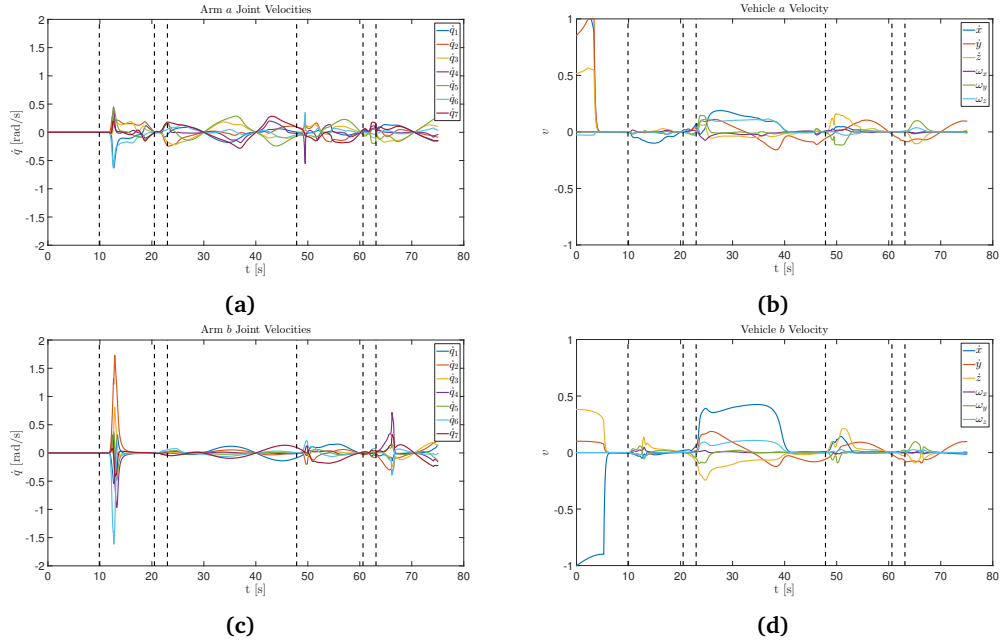


Figure 5.5: Second trial, showing that the system does not suffer from discontinuities or chattering phenomena, except at the activation or deactivation of the firm grasp constraint. The vertical dashed lines separate the different mission phases. Time history of (a) arm a joint velocities $\dot{q}$ (b) vehicle b velocities v , (c) arm b joint velocities $\dot{q}$ and (d) vehicle b velocities v .

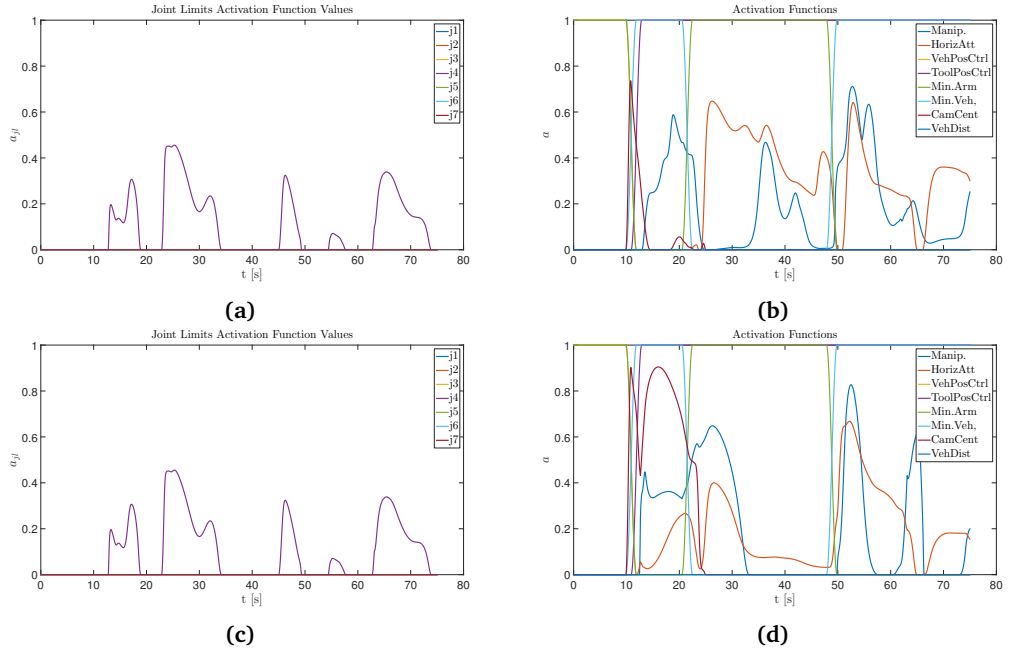


Figure 5.6: The values of the activation functions during the second simulation trial. Time history of (a) arm a joint limits activation values and (b) the activation values of all the other tasks for UVMS a (c) arm b joint limits activation values and (d) the activation values of all the other tasks for UVMS b .

5.2 Ground Base Cooperative Transportation

To finally validate the proposed architecture the control framework was tested on real hardware using a pair of YouBots by KUKA. In the previous section we focused on various action suitable for approaching and grasping an object. Here in the experimental results we will focus more on the actual multi-agent coordination action and all the data exchange involved.

We can then define the KCL configuration variables for the single mobile mobile manipulator as follows:

- The system configuration vector $\mathbf{c} \in \mathbb{R}^n$ as

$$\mathbf{c} \triangleq \begin{bmatrix} \boldsymbol{\eta} \\ \mathbf{q} \end{bmatrix}, \quad (5.2)$$

where $\boldsymbol{\eta} \in \mathbb{R}^m$ is the vehicle position and $\mathbf{q} \in \mathbb{R}^l$ is the arm configuration vector. The two agents are composed by a 3 DOF base, so to speed up calculation we can avoid putting unnecessary zeros in the configuration vector and directly defining $\boldsymbol{\eta} = [\theta_z \ x \ y]$, and a 5 DOF manipulator. Let's note that anyway we could have used a complete 6 dimensional vector and added

the additional task for underactuated vehicles as in 2.7. In this case we then have $n = m + l = 8$.

- The system velocity vector $\dot{\mathbf{y}} \in \mathbb{R}^n$ as

$$\dot{\mathbf{y}} \triangleq \begin{bmatrix} \mathbf{v} \\ \dot{\mathbf{q}} \end{bmatrix}, \quad (5.3)$$

where $\mathbf{v} \in \mathbb{R}^m$ is the vehicle velocity vector and $\dot{\mathbf{q}} \in \mathbb{R}^l$ is the arm's joint velocities vector.

For the particular organization of the joints on this mobile manipulator, even if the composition of arm plus vehicle makes up to 8 DOF, the operational space manifold is not so wide.

In this setup it has been designed a simpler task priority list with respect to the previous case, since the main focus has been to test the actual feasibility of the experiment with all the communications involved. The list of tasks is then the following, having in descending priority order:

- $\mathbf{o}_1)$ $[\mathbf{r}, \lesseqgtr, \mathbf{S}]$ Arm joints limits
- $\mathbf{o}_2)$ $[\mathbf{r}, \lesseqgtr, \mathbf{S}]$ Arm obstacle avoidance
- $\mathbf{o}_3)$ $[\mathbf{r}, \lesseqgtr, \mathbf{S}]$ Arm manipulability
- $\mathbf{o}_4)$ $[\mathbf{r}, =, \mathbf{A}]$ Tool frame position control
- $\mathbf{o}_5)$ $[\mathbf{r}, =, \mathbf{A}]$ Tool frame orientation control

where all the tasks have been already introduced in the previous sections. As usual, an additional vehicle velocity constraint is added on top of the hierarchy each run for the vehicle-arm coordination task explained in section 2.8.

5.2.1 Experimental Results

In Figure 5.7 we can see an overview of the complete architecture involved. Let's briefly go through its main elements. The setup used to test the proposed algorithm is composed by the two mobile manipulators. The two agents operate in a motion capture area, which is used to assess their absolute position, and to compute the transformation matrices necessary for setting their tool-frames $\langle t_a \rangle \equiv \langle t_b \rangle \equiv \langle o \rangle \triangleq \langle t \rangle$, as seen in Figure 4.1. The motion capture is an OptiTrack™ system consisting of a set of 8 *Flex3* cameras. On both YouBots a set of 6 markers made possible

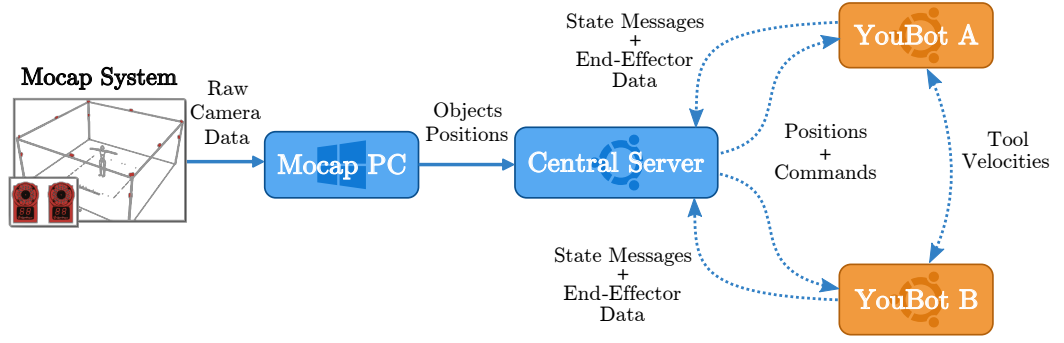


Figure 5.7: YouBot Setup overall scheme, where the arrows indicate the UDP data flow, via wired (solid) and wireless (dotted) connection.

(a) (b)

(c) (d)

(e) (f)

Figure 5.8: In these figures we can see: in 5.8a the input generated sent to the robots from the central server; in 5.8b the manipulability value for both the YouBots (where we can see a higher activation value for the manipulability task for YouBot B), in 5.8c and 5.8d the independently generated tool frame velocity on the first run on the TPIK (where we can now directly see that YouBot B is the one having more difficulties in following the reference), in 5.8e the actual resulting cooperative velocity, in 5.8f the tracking error for both the agents.

defining two rigid bodies in the dedicated *Motive* software. These rigid bodies are streamed via UDP to a central computer which is in charge of sending the relevant data to the two agents. The central server also serves the purpose of being a unified console, for controlling the two agents as a unique entity, or separately, following the modularity philosophy mentioned in the Introduction. The motion capture was deemed necessary, so to test the performances of the control architecture in a sort of isolated way. In particular without introducing too much odometry error, which is especially true for the type of wheels which the YouBot platform uses.

In the experiments, the object reference velocity $\dot{\mathbf{x}}_t$, shown in Fig. 5.8a, is generated online by a user through a dedicated software. Figure 5.8 shows an excerpt of a full experiment. In particular, at the start of the graphs $t = 75s$ the two mobile manipulators are in a configuration close to the one represented in Fig. 4.1, where the b agent is close to a singular configuration. This is shown by the fact that the manipulability task's activation function is greater than zero, as depicted in Fig. 5.8b. From $t = 80s$ to $t = 100s$ the operator specifically gives a desired object reference velocity which agent b cannot track exactly. This fact can be seen

(a) (b) (d)

Figure 5.9: In figures 5.9a and 5.9b the joint and vehicle reference velocities for YouBot A, while in figures 5.9c and 5.9d the same quantities for YouBot B.

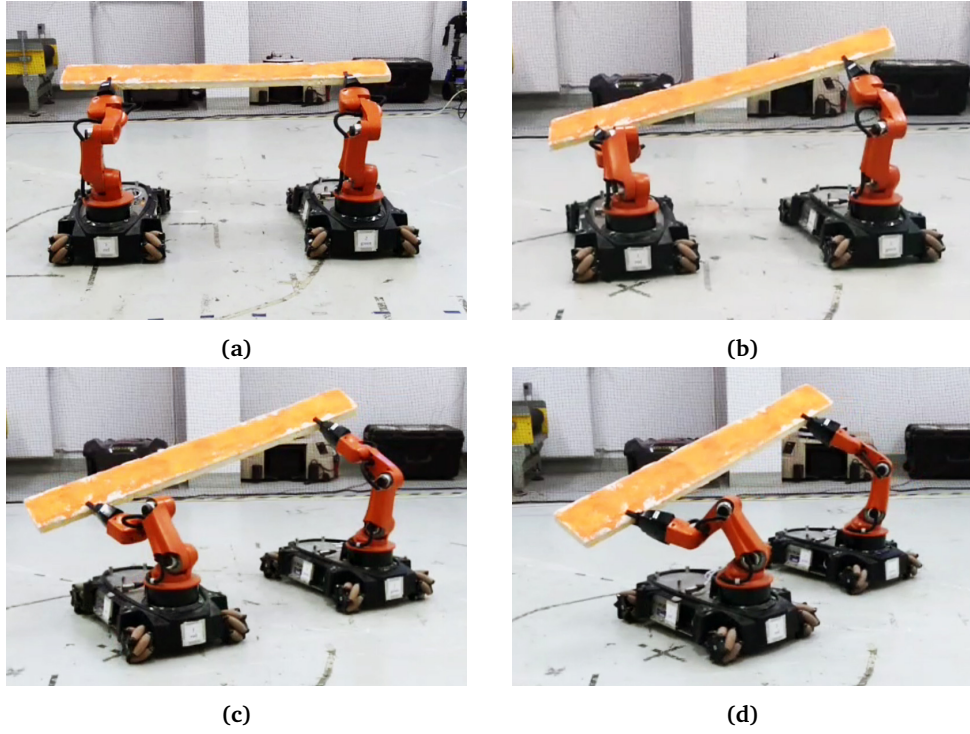


Figure 5.10: A series of successive still frames from one of the coordinated multi-robot transportation experiments. Here the robots are firstly applying a rigid translation of the object forward and then performing some rotations around the x and z axis of the object.

comparing Fig. 5.8a with Fig. 5.8c and Fig. 5.8d for agents a and b respectively, and is highlighted by the error norms $\|\dot{\mathbf{x}}_t - \dot{\mathbf{x}}_{t,i}\|$ shown in Fig. 5.8f. As can be seen in Fig. 5.8e, the *feasible cooperative* tool frame velocity is tuned in favour of agent b . Indeed, components on the linear z -direction are now present, which instead were zero in the original reference velocity. With such a coordination policy, agent b manages to keep its manipulability above the required threshold, as confirmed by the fact that its activation value in Fig. 5.8b never reaches the value one, while both systems instantaneously generate system velocities complying with the firm grasp object kinematic constraint.

CONCLUSIONS

IN this thesis a modular and expandable control framework for mobile manipulators has been presented.

Firstly, in the *Introduction* chapter the framing and the rationale behind this work have been explained. The main reasoning lays its basis on the identification of the need for a unifying control architecture for robotic mobile manipulators. Particular focus has been put on two main necessities: equipping this architecture with a series of capabilities that fit most of the needs for robotic control, with the possibility of expanding it via the definition of additional tasks; provide the flexibility for adapting it to different robotic structures and for single or multiple agent cooperation. At the end of the introduction the scientific and technological objectives highlighted six major contributions included in this work, which have been presented through the chapters. Then, in the *Experimental Outline* section we briefly described the experimental trial path followed in these years which, through incremental complexity steps, led to the successful implementation of the cooperative multi-agent experiment.

The mathematical foundations for the control algorithms have been introduced in the *Single Agent Control: Theory* chapter, giving the basic definitions for the task priority framework and completing the chapter with some expansion for mobile robot control such as the vehicle-arm coordination scheme and the multi-rate control. Of particular importance, the definition of action enabled as to take a step further hierarchy-based control, laying the basis for automatic context dependent control.

After presenting all the necessary definitions, the chapter *Single Agent Control: Applications* introduced the first experimental set trials. Starting from the simple case of single arm control, the first newly introduced tasks were defined, and the experiments proved the effectiveness of the proposed strategy. Then we moved on

a dual-arm scenario, in the more complex context of human robot collaboration. Here the task defined at the kinematic level, such as obstacle avoidance for example, showed the potential of the control architecture to relieve the higher level layer of control from dealing with low level planning, by embedding reactive strategies in the KCL. Finally, the possibility of easily scaling to a floating mobile manipulator system was showed in the last section, where we also integrated force regulation task at the kinematic level. Right now a series of on-going works are being carried out to expand the capabilities of this architecture. A first line of research is focusing on the integration with AI techniques for planning and reasoning (human robot cooperation) [37]. A second line is focusing on cooperation techniques between multiple mobile robots, exploiting strategies originally developed for underwater cooperating robots [33], to implement collaborative tasks among multiple robots in a factory scenario, highlighting the kinematic layer capability of dealing with different kind of robotic structures in a unified way.

The successive chapter, *Extension to Multi-Agent: Theory*, dived back in the theoretical realm to address the challenges brought up by multi-robot coordination. The most important contribution here has been the possibility of devising of a distributed cooperation strategy. This policy operates so to obtain a final cooperative velocity which acts in favour of the system currently resulting in a weaker condition between the two. Thanks to this policy, during the transportation the stress on the object is minimized via a mechanism where there is no pre-defined leader-follower coordination and instead the leading robot is dynamically emerging on-the-fly if necessary.

The second-last chapter, *Multi-Agent Control: Applications*, showed the experimental results obtained in various scenarios for multiple agent control. Firstly the multi-robot extension was tested in a simulative environment, where, the promising results motivated an experimental campaign on real hardware. Even if from a theoretical point of view the simulative environment has a more complex structure, the effort needed for implementing the control architecture on a physical experimental setup far outweighs the first scenario. The main challenges faced with the YouBots was to setup a solid communication infrastructure to make possible the data exchange between the YouBots and the rest of the motion capture setup. Nonetheless, the work done has well paid back with a final system that eventually proved the feasibility and effectiveness of the proposed strategy.

As stated in the introduction, having in my laboratory well-established line of

research on the topic of control of marine systems, much of my effort has been to transfer the know-how of the laboratory in underwater robotics to industrial-like scenarios. For this reasons, and the fact that the first promising results for smart manufacturing research arrived in the last part of my doctorate, most of the early and mid-term dissemination has been framed in the area of oceanic engineering as in [32, 42, 54, 55, 56, 57, 58, 59]. The last works instead are more focused on manufacturing scenarios, such as the published paper [60] and two other works, one of which has been submitted for journal publication. In particular, the currently submitted papers are the following:

- K. Darvish, F. Wanderlingh, B. Bruno, E. Simetti, F. Mastrogiovanni, and G. Casalino, “Flexible human-robot cooperation models for assisted shop-floor tasks,” *Submitted to: Mechatronics Journal (Elsevier)*. **ACCEPTED**
In this paper, a sensing, representation, planning and control architecture for flexible human-robot cooperation is proposed, as a result of the joint effort between the control and AI research groups. It collects the experiments carried with for the HRI scenarios, containing the results for contribution C_2 and C_6 .
- E. Simetti, G. Casalino, F. Wanderlingh, “A Task Priority Approach to Cooperative Mobile Manipulation: Theory and Experiments”. **WORK IN PROGRESS**
This paper contains the results obtained for the multi-agent coordination experiments carried out with the YouBots, showing the task priority framework applied to industry like robots, focusing on contribution C_5 .
- E. Simetti, G. Casalino, F. Wanderlingh, M. Aicardi. “Task Priority Control of Underwater Intervention Systems: Theory and Applications”, *Submitted to: Oceanic Engineering*. **ACCEPTED**
This work represents the final and most complete end point for task priority strategies applied to underwater robotics, collecting all the results obtained in over a decade of research.

In parallel to the theoretical work, a strong effort has been put in developing a modular and reusable software architecture. Putting all the theoretical results together in a comprehensive software package that was not just a big cauldron of algorithms was quite a challenge. Indeed, implementing a set of composable and configurable modules, the results achieved led to the development of a scalable

software architecture that was used for all the experimental trials with minimal adaptation effort for the different setups used. More details on the proposed software architecture are presented in the appendix chapter.

Future Work

Even with all the effort one can put in, there's always room for improvement. For what concerns the cooperative multi-robot experiments it would be nice to assess the quality of the experiments in relation to the coordination policy and integrate some force control for reducing the mechanical stress on the object. Also, it would be very interesting to setup an experiment, even in a simulation environment, where the coordination is performed between 3 or more agents at the same time. This, for example, could open new possibilities for the logistic industry and in general enforce the reconfiguration abilities of the proposed system. Talking about reconfigurability another possible improvement is the integration of heterogeneous robotic setups to build complex factory like scenarios. Looking forward, a highly automated factory of the future environment would see something like a pair of ground mobile manipulators handing a working piece to a dual arm robot which operates on it using some extra fixtures, maybe provided by a passing by UAV.

Another useful improvement could be done in the area of human-robot collaboration scenarios, in the direction of a better integration with the higher levels of planning, a good integration being what really makes the difference in terms of innovation, performance, flexibility and dependability. A more extensive experimental trial campaign, with the definition of complex actions and scenarios, would be a good landmark that will surely draw the attention of the industrial sector.

THE SOFTWARE ARCHITECTURE

IN parallel to the theoretical work, to make the experimental phase possible, a critical effort has been spent on the software definition and implementation for the real-time control architecture. The work started from the previous one developed during the TRIDENT project. However, its limitations and the need to use a new, recently developed real-time robotic middle-ware, have required a new implementation. Given these premises, the new architecture had the important goal to further improve the reuse of each software components. Toward that end, the methodology that I have followed is to subdivide the architecture in different processes, each one with a specific concern (e.g. Driver, Controller, Logger). The implementation of the RT framework has been done on top of the robotic middle-ware developed at GRAAL (the Ortos RT Framework). However, to facilitate the interaction with other software components that do not depend on real-time requirements, a ROS interface was also built to act as a bridge between the RT and non-RT environments. The software includes single and bimanual arm control, coordinated object transportation, leader/follower path following, all running through the task priority control module. The above-mentioned framework is structured in a series of dedicated processes, communicating via internal API or through a ROS interface. Having to deal with two task schedulers at the same time (Ortos and ROS), the integration of the two has not been so trivial, and many hours were spent in fighting with segmentation faults and broken processes, but that's a recurring old story. The architecture can be summarized as shown in A.1: each box corresponds to a process, following a modular design to ensure re-usability and real-time performances dependability.

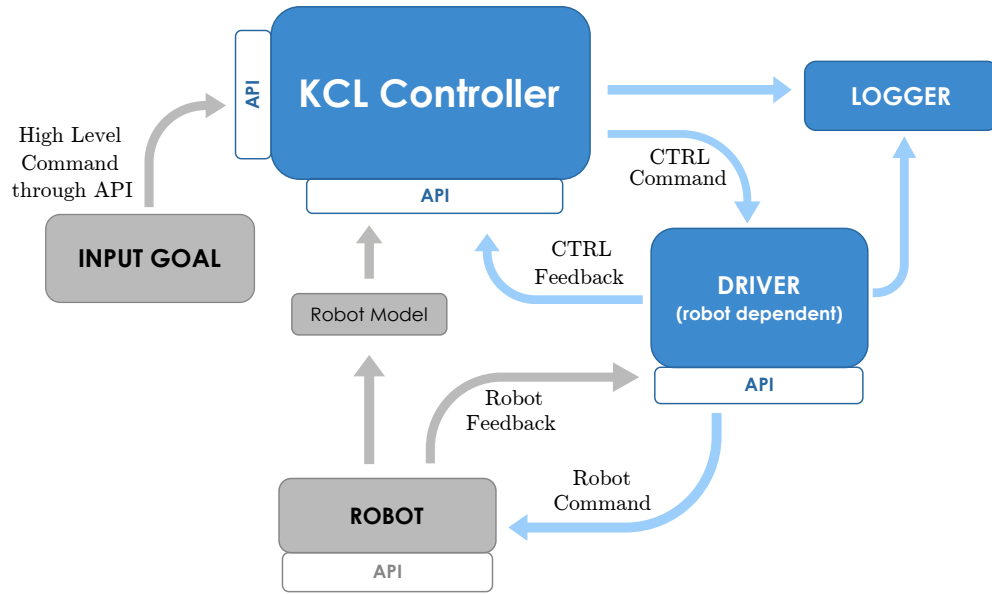


Figure A.1: A simplified overview of the developed software where three central modules of the control architecture have been highlighted in blue: the Controller, the Driver and the Logger.

A.1 Controller Modules

As above mentioned the software architecture is composed by different modules that run as soft real time tasks of the Ortos RT middle-ware scheduled in a round-robin fashion at a 100Hz frequency rate. The Ortos middle-ware provides all the utilities to implement message queues and shared memory read/write operations. In figure A.1 it is showed a simplified overview of the overall architecture, with some of its central modules being highlighted. Let's now see a little bit more in details how they operate. The nine tasks that have been developed for creating the software architecture are the following:

- ②- *KCL Controller*: the hearth of the software architecture, where all the heavy maths are going on.
- ②- *Driver*: the direct link between the software and the robot's hardware
- ②- *Logger*: listens for all the data-exchanged among the modules and stores all the variable of interest
- ②- *Data Monitor*: a collection of all the relevant system variable to check the system status at a glance
- ②- *Command Console*: a command line interface to generate simple commands

- ②- *Keyboard Controller*: an interface to generate references for the controller in a tele-operated fashion
- ②- *UDP Receiver*: the module where data can be received from other computers running the software architecture
- ②- *UDP Sender*: the module where data can be sent to other computers running the software architecture
- ②- *MoCap Bridge*: the link between the motion capture system and the control architecture

Let's now spend just few words going through these nine modules and seeing how they relate one another.

KCL Controller

This first process is the principal control module, where all the information about the actual state of the system is gathered, and the task priority inverse kinematics takes place. It is the end-point for the controller interface, that has been implemented in the `ControllerInterface` class. This class exposes all the functionalities of the framework for an easy integration of the KCL with the HLL, thereby enforcing the C_6 contribution. This module loads the model of the robot at hand and stores it in the `ArmModel` class. The model smart pointer is passed through classes so to always have an updated status of the model available across the whole control architecture. It communicates mainly with the *Driver* from which it receives the feedback and other state relevant variables such as: enable status of the robot, availability of grippers, status of the calibration and more. After all the model loading and other bootstrap procedures, the task When a command has been issued by the control interface, once the control has been computed (via the `MultiArm` class), it sends the control reference back to the *Driver*. This module also provides an alive signal which tells whether the process is actually running and if the initial bootstrap has been successful. This information is used by other modules to provide relevant warnings and also in the graphical user interface (GUI) which is presented in section A.2.

Driver

The driver is the module which directly communicates with the robot and exposes an interface which makes the control of the robot transparent to all the other modules, the *Controller* in particular. It can have internal different implementations depending on the way we access the robot actuators and sensor. Let's take some examples based on the Baxter and YouBot platform. On the Baxter we have no access to the robot internal hardware due to proprietary restrictions, and this module is implemented as a ROS wrapper. In the YouBot case instead we had access to the hardware APIs, and in the driver all the base and gripper calibration had to be implemented. The *Driver* is mainly composed of the `Robot::Driver` class which is subdivided in the `Robot::Base` and `Robot::Arm` classes. Also this module provides an alive signal which tells whether the process is actually running and if the initial robot procedures have been successful.

Logger

The logging module is one the most important modules, as without it our experiment would just be overly complicated diversions. This task in particular is the only for which a slower scheduling rate has been chosen, due to the high input/output operations throughput. This was particularly the case for the Baxter case where, having two 7 DOFs arms we have a massive amount of data to be saved each iteration. All the data is read from the Ortos shared memory and includes: robot parameters, robot feedback (position, velocity, control mode), references evaluated by the controller, task priority relevant variables for each task (intermediate rhos, activations, enable). Using the console it is possible to start and stop the logging.

Data Monitor

The *Data Monitor* shows a collection of all the relevant system variables to check the system status at a glance, reading all the data from the Ortos shared memory written by the other tasks. It has been of great help during the development of the code to check what was going on when the system was acting like crazy for some insidious error, which as you would expect happened 80% of the time.

Command Console

The *Command Console* is command line interface to generate simple commands to control the robot. It can be very useful when testing single functionalities of the framework during implementation, but can also serve as a control interface for simple tasks. Via this modules you can access most of the capabilities offered by the controller. It has been developed both as interactive shell and as a terminal CLI (to provide command history). Just as an example, the typical screen for the command console can be the following:

```
jointPos      < arm j1 ... jn >
jointVel      < arm j1 ... jn >
cartPos       < arm y p r x y z >
cartRelToolPos < arm y p r x y z >
cartVel       < arm y p r x y z >
cartVelInput  < arm >
stopArm       < arm >
openGrip      < arm >
closeGrip     < arm >
reloadPar     < arm >
holdMode      < arm [0(disable)|1(enable)] >
startLog      < arm >
stopLog       < arm >
unfoldArm     < arm >
foldArm       < arm >

> Type a command...
```

In the interactive shell version of the console also, all the messages issued by the other modules are displayed in a unified way, clearly identified with a sender name and colour, to have a single place where to look for knowing the system's state.

UDP Sender and Receiver

The *UDP Sender* module, together with *UDP Receiver* one, became necessary for the realization of the YouBots experiments, where multiple PCs had to communicate one another, as summarized earlier in Figure 5.7. Depending on a series of input parameters these module can be configured to run for reading agent related inputs (as on the YouBots) or external coordination data (as in the Central Server of the above mentioned figure). The receiver module 'simply' opens a series of non-blocking UDP sockets and writes the received data to the Ortos framework. The sender act in a complementary way.

Keyboard Controller

This task is a simple keyboard controller that can be used to generate references for the KCL in a tele-operated fashion. It makes use of the SDL Library, a well known library for input management, that gives the possibility to control the robot in a WASD-like mode. Via this controller is possible to switch between sending smoothly increasing and decreasing references (implemented with a simple first order filter), or step-wise references.

MoCap Bridge

The motion capture task serves as a bridge between the OptiTrack™ system and the control framework. In particular, it reads the rigid body data streamed by the MoCap proprietary software and translates it into coordinate frames compatible with the mathematical library. The data is then sent to the *UPD Receiver* tasks of each agent. One of the main issues has also been the correct axis interpretation, as everyone uses its own *standard*, and getting it right isn't always an easy task.

Let's note how in the last YouBot experiments for example, where we have multiple PCs composing the overall system, the architecture running is the same for all the sub-systems. Of course the controller and driver modules will be only running on the agents, but the general software structure is shared, with just different configuration parameters being parsed. Also the definition of a standardized API interface for such processes allows the architecture to be as much as possible generic and reusable for different setups.

A.1.1 External Libraries

To make the controller development possible, a series of external libraries as been included. These are all the libs that have been used:

- **Ortos** framework - Realtime middleware
- **ROS Indigo** - The ROS Operating system
- **CMAT** - Robotics mathematical library
- **Libconfig 1.5** - Configuration file manager
- **SDL 1.2** - Media library for user input
- **Beep** - Utility program for acoustic signalling

To improve the usability of the software, a GUI interface has been developed to interact with the controller whose details are included in the appendix.

A.2 Graphical User Interface

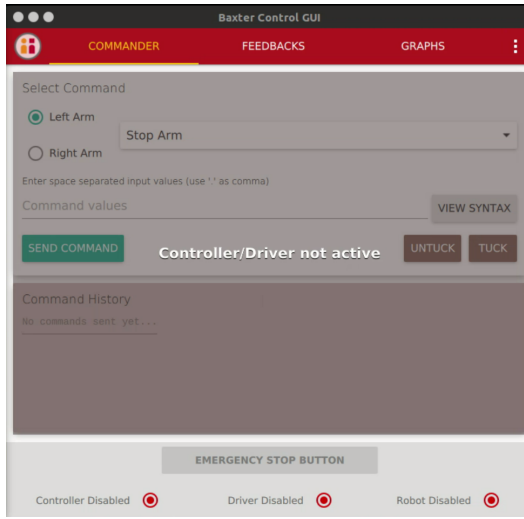
During the PhD work I also developed a simple user interface. The reasons that lead to this decision are fundamentally three: a personal interest in interface design, speed up the testing phase without having to hassle with too many open terminals (and whoever works in applied computer science knows what I mean), and last but not least providing a tool that was easy to use and fool's proof for whoever had the need of using such framework. The interface was developed in QT5 using the new QML scripting language. QML is basically a JavaScript-like language for developing cross-platform window applications and Model-View-Controller relationships.

It all started with a simple EMERGENCY-STOP button to stop the robot in case of critical errors, and lately I expanded the graphical interface to encompass the following features:

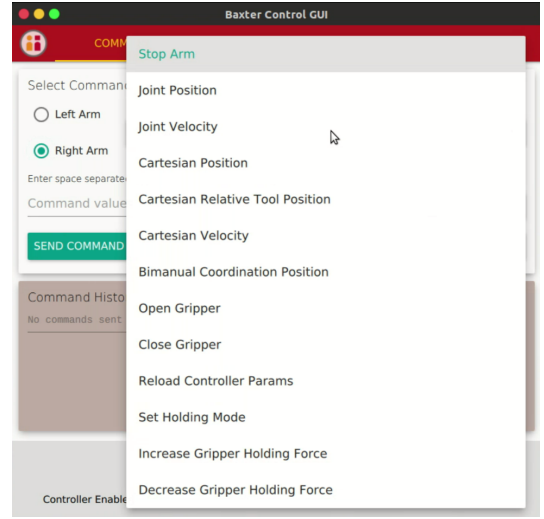
- Enable/Disable the robot
- Provide status indicators
- Sending control commands (with type checking)
- View Command History
- View robot feedbacks
- Plot real-time graphs (still in beta phase)

The interfaces were developed for the Baxter and the Youbot robots. They share the same code base, apart from some robot dependent customization. In Figure A.2 are showed some self-explanatory screenshots of the interface for the Baxter Robot.

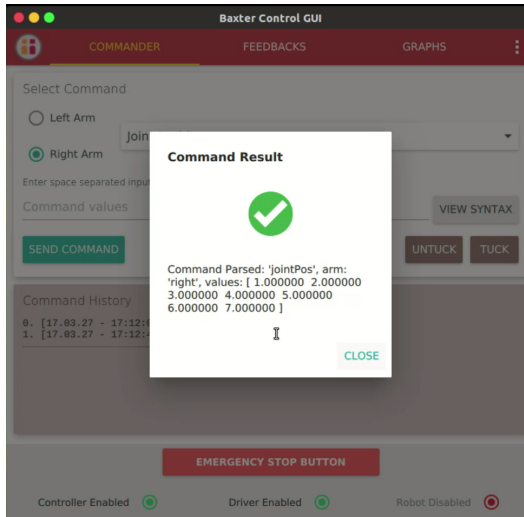
Let's briefly describe the various states of the interfaces showed in Figure A.2. The interface is composed of three main tabs: Commander, Feedbacks and Graphs. The original Emergency-Stop button is very wide and always accessible at the bottom of the interface, so to be functional even in case of a critical situation. Together with the stop button at the bottom of the interface we find the system status indicators for: the controller, the driver and the robot. In figure A.2a we can see the interfaces when no active controller is detected in the system, note all the red indicators in the bottom. In the other figure instead the Driver and Controller have been



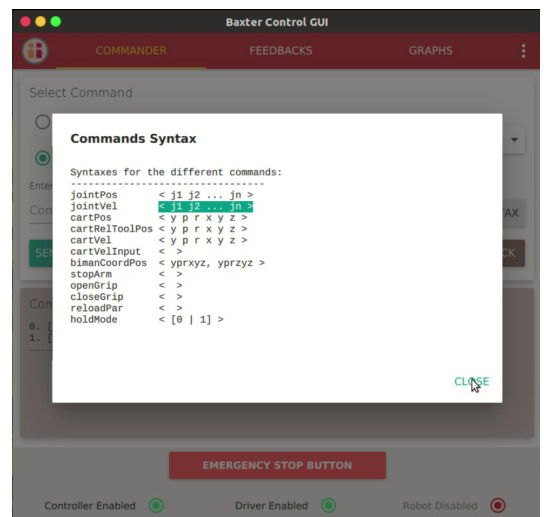
(a)



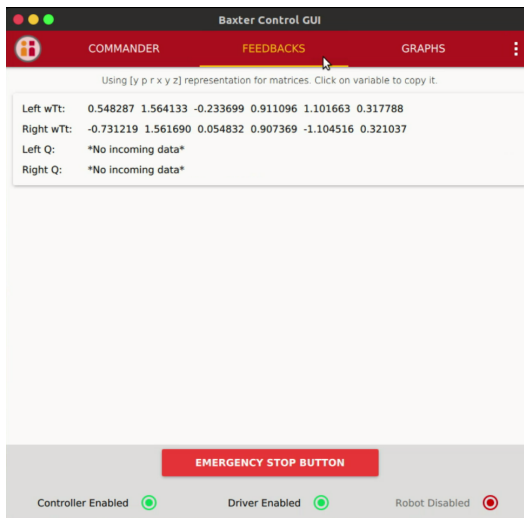
(b)



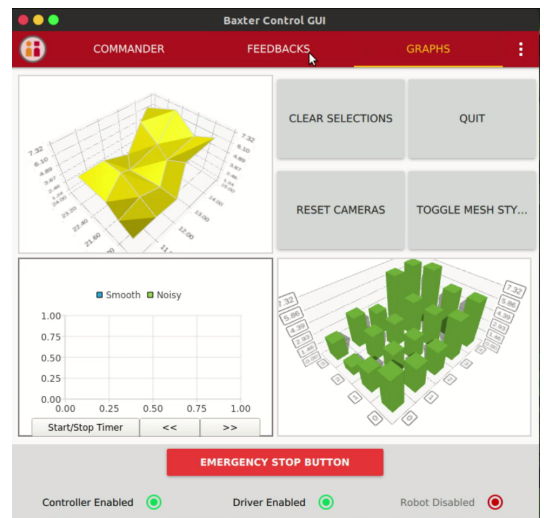
(c)



(d)



(e)



(f)

Figure A.2: Various states and tabs of the GUI.

activated, the status indicators became green and the interface becomes active. Figure A.2b shows the available commands to be sent. The interface is equipped with a syntax spelling checker which informs the user if a sent command was meaningful or not, informing it with a pop-up message. In figure A.2c we can see the message resulting from a correct parsing of the command, whose syntax is available with the button VIEW SYNTAX on the right (as seen in A.2d). Figure A.2e shows the feedback tab. Figure A.2f instead shows the prototype tab for plotting related features.

LIST OF FIGURES

1.1	A focused extract from the Italian “ <i>Research and Innovation Roadmap</i> ” [3] for smart manufacturing, highlighting the scope in which this PhD work is framed.	2
1.2	Basic structural robotic elements in figures 1.2a and 1.2b, and some of their possible compositions. In figures 1.2d and 1.2e is highlighted a Single-agent vs. Multi-agent control.	7
1.3	General overview of the robot architectures that have been successively investigated. . .	12
2.1	The overall architecture: the Kinematic Control Layer is the one implementing the proposed task priority approach, executing the current action scheduled by a Mission Manager; the output of the Kinematic Control Layer are the system velocities, tracked by the underlying Dynamic Control Layer.	15
2.2	Comparison of task activations for equality and inequality control objectives. In figure 2.2b and example of $\sigma_{\downarrow}(x_{min}, x_{min} + \Delta)$ decreasing sigmoidal function.	22
2.3	A detailed view of the Kinematic Control Layer, with the two Task Priority Inverse Kinematics blocks highlighted, performing the vehicle velocity tracking compensation scheme in presence of an input action $\mathcal{A}$	29
3.1	Single arm experimental trial: (a) activation functions for the joint limits task, (b) values of the joint near its physical limit (c) time history of the joint velocities during the trial.	36
3.2	Frames involved in the rigid grasp manipulation: the left and right end-effector frames $\langle t_L \rangle$ and $\langle t_R \rangle$, and the object frame $\langle o \rangle$	38
3.3	The three-layered architecture. In the HLL module the Scene Perception module sends the information on the current entities present in the scene to the Path Planning module which then communicates with the Mission Control module to interact with the kinematic layer. The KCL translates the commands received to control objectives and produces the reference joint velocities for the underlying dynamic layer. The DCL follows the given velocity at its best sending the final commands to the Joint Control Boards (JCB).	41
3.4	A series of successive still frames from one of the coordinated transportation experiments. As it can be seen in the last two frames 3.4c and 3.4d, the elbow is gradually raising its height to stay out of the bounding box defined for the obstacle.	42
3.5	In figure 3.5f scene and objects recognized by OSP and relative frame of the vision system, each colour is identifying a different object. Figures 3.5a and 3.5b are for the experiments with obstacle avoidance only at the HLL level and figures 3.5c, 3.5d and 3.5e are for the experiments also with KCL reactive avoidance.	43
3.6	The HRI experiment where an obstacle is detected and avoided by the elbow joint while bi-manually holding a plate that’s used by a human operator. The safety threshold has been highlighted in green. In figure 3.6d it can be observed how the right robot’s forearm is higher than the left forearm which is avoiding the library.	44

3.7	Elbow distance from the obstacle and activation function of the elbow avoidance task for the right arm: in <i>blue</i> for a single-arm operation, in <i>orange</i> for a dual-arm operation.	45
3.8	Screenshots of the UVMS as it performs the pipe welding inspection task.	48
3.9	Simulation results: (a) the time behaviour of the norm of the force (reference value is 10 N), (b) the arm joint velocities (saturation value at 1 rad/s) (c) vehicle reference velocities (saturation value at 0.2 rad/s and 0.2 m/s) (d) the activation values of the joint limit tasks, (e) the activation values of the other tasks, (f) reference path (red) and actual trajectory (blue).	50
4.1	Frames involved in the cooperative mobile manipulation: the vehicle frame $\langle v \rangle$, the end-effector frames $\langle e \rangle$ and the tool control point $\langle t \rangle$ which coincides with object frame $\langle o \rangle$	53
4.2	Block diagram of the cooperation, as seen from agent a . 1) a TPIK (action $\mathcal{A}_m$) is first solved, acting as if the agent was alone 2) the system velocity vector output of the TPIK is used to compute the <i>non-cooperative</i> tool-frame velocity 3) the non-cooperative tool frame velocity is processed according to the given <i>Coordination Policy</i> , which computes the <i>feasible cooperative</i> one through (4.7) and (4.9) 4) Agent a now solves a new TPIK (action $\mathcal{A}_c$) constrained to the feasible cooperative tool-frame velocity	54
5.1	The UVMS and its relevant frames	63
5.2	Screenshots of the simulation environment as the transportation mission is executed: (a) initial position $t = 0$ (b) successful grasp of the objects by both UVMS $t = 13.71$ (c) transportation toward the goal position $t = 20$ (d) pre-deployment position reached $t = 39.3$ (e) deployment of the pipe $t = 45$ (f) deployment completed $t = 55$	65
5.3	The system velocities during the first simulation trial, showing that the system does not suffer from discontinuities or chattering phenomena, except at the activation or deactivation of the firm grasp constraint. The vertical dashed lines separate the different mission phases. Time history of (a) arm a joint velocities $\dot{\mathbf{q}}$ (b) vehicle b velocities $\mathbf{v}$, (c) arm b joint velocities $\dot{\mathbf{q}}$ (d) vehicle b velocities $\mathbf{v}$ (e) a zoom of the arm b joint velocities and (f) of the vehicle b velocities in the interval $38 < t < 41$ to show the continuity of the approach.	67
5.4	First trial. Time history of (a) arm a joint limits activation values and (b) the activation values of all the other tasks for UVMS a (c) arm b joint limits activation values and (d) the activation values of all the other tasks for UVMS b	68
5.5	Second trial, showing that the system does not suffer from discontinuities or chattering phenomena, except at the activation or deactivation of the firm grasp constraint. The vertical dashed lines separate the different mission phases. Time history of (a) arm a joint velocities $\dot{\mathbf{q}}$ (b) vehicle b velocities $\mathbf{v}$, (c) arm b joint velocities $\dot{\mathbf{q}}$ and (d) vehicle b velocities $\mathbf{v}$	68
5.6	The values of the activation functions during the second simulation trial. Time history of (a) arm a joint limits activation values and (b) the activation values of all the other tasks for UVMS a (c) arm b joint limits activation values and (d) the activation values of all the other tasks for UVMS b	69
5.7	YouBot Setup overall scheme, where the arrows indicate the UDP data flow, via wired (solid) and wireless (dotted) connection.	71
5.8	In these figures we can see: in 5.8a the input generated sent to the robots from the central server, in 5.8b the manipulation value for both the YouBots (where we can see a higher activation value for the manipulability task for YouBot B), in 5.8c and 5.8d the independently generated tool frame velocity on the first run on the TPIK (where we can now directly see that YouBot B is the one having more difficulties in following the reference), in 5.8e the actual resulting cooperative velocity, in 5.8f the tracking error for both the agents.	71

5.9	In figures 5.9a and 5.9b the joint and vehicle reference velocities for YouBot A, while in figures 5.9c and 5.9d the same quantities for YouBot B.	72
5.10	A series of successive still frames from one of the coordinated multi-robot transportation experiments. Here the robots are firstly applying a rigid translation of the object forward and then performing some rotations around the x and z axis of the object.	72
A.1	A simplified overview of the developed software where three central modules of the control architecture have been highlighted in blue: the Controller, the Driver and the Logger.	78
A.2	Various states and tabs of the GUI.	84

REFERENCES

- [1] SPARC The Partnership for Robotics in Europe, *Robotics 2020 Multi-Annual Roadmap*. eu-Robotics AISBL, 2015.
- [2] European Commission. (2017, Mar.) ICT-25-2016-2017, "Advanced robot capabilities research and take-up". [Online]. Available: <https://goo.gl/f3LT1u>
- [3] Gruppo Roadmapping CFI, *Roadmap per la ricerca e l'innovazione - Cluster Fabbrica Intelligente*. Associazione Cluster Fabbrica Intelligente, 2015. [Online]. Available: <https://goo.gl/VNatYo>
- [4] G. Cheng, S. Hyon, J. Morimoto, A. Ude, J. G. Hale, G. Colvin, W. Scroggin, and S. C. Jacobsen, "Cb: A humanoid research platform for exploring neuroscience," *Advanced Robotics*, vol. 21, no. 10, pp. 1097–1114, 2007.
- [5] C. Borst, T. Wimbock, F. Schmidt, M. Fuchs, B. Brunner, F. Zacharias, P. R. Giordano, R. Konietschke, W. Sepp, S. Fuchs, C. Rink, A. Albu-Schaffer, and G. Hirzinger, "Rollin' justin - mobile platform with variable base," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 1597–1598.
- [6] M. A. Diftler, J. S. Mehling, M. E. Abdallah, N. A. Radford, L. B. Bridgwater, A. M. Sanders, R. S. Askew, D. M. Linn, J. D. Yamokoski, F. A. Permenter, B. K. Hargrave, R. Platt, R. T. Savely, and R. O. Ambrose, "Robonaut 2 - the first humanoid robot in space," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 2178–2183.
- [7] J. Fink, N. Michael, S. Kim, and V. Kumar, "Planning and control for cooperative manipulation and transportation with aerial robots," *International Journal of Robotics Research*, vol. 30, no. 3, pp. 324–334, 2011.
- [8] E. Simetti, A. Turetta, and G. Casalino, "Distributed control and coordination of cooperative mobile manipulator systems," in *Distributed Autonomous Robotic Systems 8*, H. Asama, H. Kurokawa, J. Ota, and K. Sekiyama, Eds. Springer Berlin Heidelberg, 2009, pp. 315–324.
- [9] G. Marani, S. K. Choi, and J. Yuh, "Underwater autonomous manipulation for intervention missions AUVs," *Ocean Engineering*, vol. 36, pp. 15–23, 2008.
- [10] P. Sanz, R. Ridao, G. Oliver, P. Casalino, C. Insaurralde, C. Silvestre, M. Melchiorri, and A. Turetta, "Trident: Recent improvements about autonomous underwater intervention missions," in *Proceedings of the IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles (NGCUV2012)*, Porto, Portugal, 2012.
- [11] P. Tsarouchi, S. Makris, G. Michalos, M. Stefanos, K. Fourtakas, K. Kaltsoukalas, D. Kontrovakis, and G. Chrysosolouris, "Robotized assembly process using dual arm robot," *Procedia CIRP*, vol. 23, no. Supplement C, pp. 47 – 52, 2014, 5th CATS 2014 - CIRP Conference on Assembly Technologies and Systems. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S2212827114011354>

-
- [12] S. Makris, G. Michalos, A. Eytan, and G. Chryssolouris, "Cooperating robots for reconfigurable assembly operations: Review and challenges," *Procedia CIRP*, vol. 3, no. Supplement C, pp. 346 – 351, 2012, 45th CIRP Conference on Manufacturing Systems 2012. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S2212827112002326>
 - [13] J. Saenz, F. Penzlin, C. Vogel, and M. Fritzsche, "VALERI – A collaborative mobile manipulator for aerospace production," in *Advances in Cooperative Robotics*. World Scientific, 2017, pp. 186–195.
 - [14] M. R. Pedersen, L. Nalpantidis, R. S. Andersen, C. Schou, S. Bøgh, V. Krüger, and O. Madsen, "Robot skills for manufacturing: From concept to industrial deployment," *Robotics and Computer-Integrated Manufacturing*, vol. 37, pp. 282–291, 2016.
 - [15] F. Caccavale and M. Uchiyama, "Cooperative manipulation," in *Springer Handbook of Robotics*. Springer, 2016, pp. 989–1006.
 - [16] T. G. Sugar and V. Kumar, "Control of cooperating mobile manipulators," *IEEE Transactions on robotics and automation*, vol. 18, no. 1, pp. 94–103, 2002.
 - [17] A. Yamashita, T. Arai, J. Ota, and H. Asama, "Motion planning of multiple mobile robots for cooperative manipulation and transportation," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 2, pp. 223–237, 2003.
 - [18] N. Miyata, J. Ota, T. Arai, and H. Asama, "Cooperative transport by multiple mobile robots in unknown static environments associated with real-time task assignment," *IEEE transactions on robotics and automation*, vol. 18, no. 5, pp. 769–780, 2002.
 - [19] A. Yufka and M. Ozkan, "Formation-based control scheme for cooperative transportation by multiple mobile robots," *International Journal of Advanced Robotic Systems*, vol. 12, no. 9, p. 120, 2015.
 - [20] Z. Wang and M. Schwager, "Force-amplifying n-robot transport system (force-ants) for cooperative planar manipulation without communication," *The International Journal of Robotics Research*, vol. 35, no. 13, pp. 1564–1586, 2016.
 - [21] H. Yamaguchi, A. Nishijima, and A. Kawakami, "Control of two manipulation points of a cooperative transportation system with two car-like vehicles following parametric curve paths," *Robotics and Autonomous Systems*, vol. 63, pp. 165–178, 2015.
 - [22] O. Khatib, K. Yokoi, K. Chang, D. Ruspini, R. Holmberg, and A. Casal, "Coordination and decentralized cooperation of multiple mobile manipulators," *Journal of Field Robotics*, vol. 13, no. 11, pp. 755–764, 1996.
 - [23] R. Holmberg and O. Khatib, "Development and control of a holonomic mobile robot for mobile manipulation tasks," *The International Journal of Robotics Research*, vol. 19, no. 11, pp. 1066–1074, 2000.
 - [24] Z. Li, P. Y. Tao, S. S. Ge, M. Adams, and W. S. Wijesoma, "Robust adaptive control of cooperating mobile manipulators with relative motion," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 1, pp. 103–116, 2009.
 - [25] A. Marino and P. Chiacchio, "Task-oriented decentralized adaptive control of cooperative manipulators," in *Robotics and Biomimetics (ROBIO), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1325–1330.
 - [26] S. Erhart, D. Sieber, and S. Hirche, "An impedance-based control architecture for multi-robot cooperative dual-arm mobile manipulation," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 315–322.

-
- [27] S. Erhart and S. Hirche, "Internal force analysis and load distribution for cooperative multi-robot manipulation," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1238–1243, 2015.
 - [28] —, "Model and analysis of the interaction dynamics in cooperative manipulation tasks," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 672–683, 2016.
 - [29] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, 2014.
 - [30] S. Moe, G. Antonelli, A. R. Teel, K. Y. Pettersen, and J. 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*, vol. 3, p. 16, 2016.
 - [31] E. Simetti and G. Casalino, "A novel practical technique to integrate inequality control objectives and task transitions in priority based control," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1, pp. 877–902, apr 2016.
 - [32] E. Simetti, F. Wanderlingh, S. Torelli, M. Bibuli, A. Odetti, G. Bruzzone, D. Lodi Rizzini, J. Aleotti, G. Palli, L. Moriello, and U. Scarcia, "Autonomous underwater intervention: Experimental results of the MARIS project," *IEEE Journal of Oceanic Engineering*, pp. 1–20, 2017.
 - [33] E. Simetti and G. Casalino, "Manipulation and transportation with cooperative underwater vehicle manipulator systems," *IEEE Journal of Oceanic Engineering*, pp. 1–18, 2016.
 - [34] G. Muscio, F. Pierri, M. Trujillo, E. Cataldi, G. Antonelli, F. Caccavale, A. Viguria, S. Chiaverini, and A. Ollero, "Coordinated control of aerial robotic manipulators: Theory and experiments," *IEEE Transactions on Control Systems Technology*, 2017.
 - [35] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *Robotics and Automation, IEEE Transactions on*, vol. 13, no. 3, pp. 398–410, 1997.
 - [36] K. Baizid, G. Giglio, F. Pierri, M. A. Trujillo, G. Antonelli, F. Caccavale, A. Viguria, S. Chiaverini, and A. Ollero, "Behavioral control of unmanned aerial vehicle manipulator systems," *Autonomous Robots*, vol. 41, no. 5, pp. 1203–1220, Jun 2017.
 - [37] K. Darvish, B. Bruno, E. Simetti, F. Mastrogiovanni, and G. Casalino, "An adaptive human-robot cooperation framework for assembly-like tasks," in *3rd Workshop on Artificial Intelligence and Robotics (AIRO 2016), hosted by the 15th Conference of the Italian Association for Artificial Intelligence (AlxIA 2016)*, 2016.
 - [38] C. Recchiuto, A. Sgorbissa, F. Wanderlingh, and R. Zaccaria, "UAV teams in emergency scenarios: A summary of the work within the project prisma," in *CEUR Workshop Proceedings*, vol. 1544, 2015, pp. 40–53.
 - [39] C. Nattero, C. Recchiuto, A. Sgorbissa, F. Wanderlingh, and R. Zaccaria, "Coverage algorithms for search and rescue with UAV drones," in *XIII AI*IA Symposium on Artificial Intelligence, AIRO Workshop*, 2014.
 - [40] G. Casalino, E. Simetti, N. Manerikar, A. Sperinde, S. Torelli, and F. Wanderlingh, "Cooperative underwater manipulation systems: Control developments within the maris project," in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 48, no. 2, 2015, pp. 1–7. [Online]. Available: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84954164708&partnerID=40&md5=aa899addfa74b4a7f0b0b8fe8cc3150b>

- [41] E. Simetti, G. Casalino, N. Manerikar, A. Sperinde, S. Torelli, and F. Wanderlingh, "Cooperation between autonomous underwater vehicle manipulations systems with minimal information exchange," in *MTS/IEEE OCEANS 2015 - Genova: Discovering Sustainable Ocean Energy for a New World*, 2015. [Online]. Available: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84957687427&partnerID=40&md5=60d53d12b8b6e20fe54c2cd5e140eac5>
- [42] E. Simetti, G. Casalino, F. Wanderlingh, and G. Indiveri, G. ans Antonelli, "Dexrov project: Control framework for underwater interaction tasks," in *Oceans'17 MTS/IEEE Aberdeen: A Vision for Sustaining Our Marine Futures*, 2017.
- [43] T. Perez and T. I. Fossen, "Kinematic models for manoeuvring and seakeeping of marine vessels," *Modeling, Identification and Control*, vol. 28, no. 1, pp. 19–30, 2007.
- [44] K. L. Doty, C. Melchiorri, and C. Bonivento, "A theory of generalized inverses applied to robotics," *International Journal of Robotics Research*, vol. 12, no. 1, pp. 1–19, 1993.
- [45] G. Antonelli, G. Indiveri, and S. Chiaverini, "Prioritized closed-loop inverse kinematic algorithms for redundant robotic systems with velocity saturations," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 5892–5897.
- [46] O. Kanoun, F. Lamiroux, and P. B. Wieber, "Kinematic control of redundant manipulators: generalizing the task-priority framework to inequality task," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 785–792, 2011.
- [47] B. Ilon, "Wheels for a course stable selfpropelling vehicle movable in any desired direction on the ground or some other base," Apr. 8 1975, uS Patent 3,876,255. [Online]. Available: <https://www.google.com/patents/US3876255>
- [48] T. Yoshikawa, "Manipulability of robotic mechanisms," *Int. J. of Robotics Research*, vol. 4, no. 1, pp. 3–9, 1985.
- [49] G. Marani, J. Kim, J. Yuh, and W. K. 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*, vol. 2, 2002, pp. 1973–1978.
- [50] D. Ribas, P. Ridao, A. Turetta, C. Melchiorri, G. Palli, J. Fernandez, and P. Sanz, "I-AUV Mechatronics Integration for the TRIDENT FP7 Project," *Mechatronics, IEEE/ASME Transactions on*, vol. 20, no. 5, pp. 2583–2592, 2015.
- [51] E. Simetti, S. Galeano, and G. Casalino, "Underwater vehicle manipulator systems: Control methodologies for inspection and maintenance tasks," in *OCEANS 2016 - Shanghai*. Shanghai, China: IEEE, apr 2016.
- [52] L. L. Whitcomb and D. R. Yoerger, "Preliminary experiments in model-based thruster control for underwater vehicle positioning," *IEEE Journal of Oceanic Engineering*, vol. 24, no. 4, pp. 495–506, 1999.
- [53] D. Ribas, N. Palomeras, P. Ridao, M. Carreras, and A. Mallios, "Girona 500 AUV: From survey to intervention," *IEEE/ASME Transactions on Mechatronics*, vol. 17, no. 1, pp. 46–53, feb 2012.
- [54] G. Casalino, E. Simetti, and F. Wanderlingh, "Robotized underwater interventions," in *Sensing and Control for Autonomous Vehicles*. Springer International Publishing, 2017, pp. 365–386.
- [55] G. Casalino, M. Caccia, S. Caselli, C. Melchiorri, G. Antonelli, A. Caiti, G. Indiveri, G. Cannata, E. Simetti, S. Torelli, A. Sperindè, F. Wanderlingh, G. Muscolo, M. Bibuli, G. Bruzzone, E. Zereik, A. Odetti, E. Spirandelli, A. Ranieri, J. Aleotti, D. Rizzini, F. Oleari, F. Kallasi, G. Palli, U. Scarcia, L. Moriello, and E. Cataldi, "Underwater intervention robotics: An outline of the italian national project maris," *Marine Technology Society Journal*, vol. 50, no. 4, pp. 98–107, 2016. [Online]. Available: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84983382271&partnerID=40&md5=32227425036e936900fc4297dcbf95f1>

-
- [56] E. Simetti, G. Casalino, F. Wanderlingh, and G. Indiveri, G. ans Antonelli, “Robust project: Control framework for deep sea mining exploration,” in *Oceans’17 MTS/IEEE Anchorage: Our Harsh and Fragile Ocean*, 2017.
- [57] E. Simetti, G. Casalino, N. Manerikar, S. Torelli, A. Sperindé, and F. Wanderlingh, “Cooperation between autonomous underwater vehicle manipulations systems with minimal information exchange,” in *OCEANS 2015 - Genova*. Genova, Italy: IEEE, may 2015.
- [58] G. Casalino, E. Simetti, N. Manerikar, A. Sperindé, S. Torelli, and F. Wanderlingh, “Cooperative underwater manipulation systems control developments within the MARIS project,” in *IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles (NGCUV 2015)*, vol. 48, no. 2. Elsevier BV, April 2015, pp. 1–7.
- [59] G. Casalino, M. Caccia, S. Caselli, C. Melchiorri, G. Antonelli, A. Caiti, G. Indiveri, G. Cannata, E. Simetti, S. Torelli, A. Sperindé, F. Wanderlingh, G. Muscolo, M. Bibuli, G. Bruzzone, E. Zereik, A. Odetti, E. Spirandelli, A. Ranieri, J. Aleotti, D. Lodi Rizzini, F. Oleari, F. Kallasi, G. Palli, U. Scarcia, L. Moriello, and E. Cataldi, “Underwater intervention robotics: An outline of the italian national project MARIS,” *Marine Technology Society Journal*, vol. 50, no. 4, pp. 98–107, jul 2016.
- [60] G. Casalino, E. Simetti, F. Wanderlingh, K. Darvish, B. Bruno, and F. Mastrogiovanni, “On autonomous robotic cooperation capabilities within factory and logistic scenarios,” *Procedia Manufacturing*, vol. 11, pp. 147–163, 2017.