Filippo Sanfilippo

ALTERNATIVE AND FLEXIBLE CONTROL METHODS FOR ROBOTIC MANIPULATORS

On the challenge of developing a flexible control architecture that allows for controlling different manipulators

Thesis for the degree of Philosophiae Doctor

Trondheim, June 2015

Norwegian University of Science and Technology
Faculty of Information Technology,
Mathematics and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Engineering Cybernetics

978-82-326-1040-2 (print)
978-82-326-1041-9 (digital)
1503-8181

Doctoral theses at NTNU, 2015:192

Printed by NTNU Grafisk senter

This work is dedicated to my parents Rosario and Concetta. All I have and will accomplish are only possible due to their love and sacrifices.

This dedication is extended also to my brother Riccardo and my sister Elisa for all their support and encouragement.

This work is also dedicated to my girlfriend Rūta who has been a great source of motivation and inspiration.

Finally, this work is also dedicated to all those who believe in the richness of learning and in the importance of scientific research.

Abstract

Robotic arms and cranes show some similarities in the way they operate and in the way they are designed. Both have a number of links serially attached to each other by means of joints that can be moved by some type of actuator. In both systems, the end-effector of the manipulator can be moved in space and be placed in any desired location within the system's workspace and can carry a certain amount of load. However, traditional cranes are usually relatively big, stiff and heavy because they normally need to move heavy loads at low speeds, while industrial robots are ordinarily smaller, they usually move small masses and operate at relatively higher velocities. This is the reason why cranes are commonly actuated by hydraulic valves, while robotic arms are driven by servo motors, pneumatic or servo-pneumatic actuators. Most importantly, the fundamental difference between the two kinds of systems is that cranes are usually controlled by a human operator, joint by joint, using simple joysticks where each axis operates only one specific actuator, while robotic arms are commonly controlled by a central controller that controls and coordinates the actuators according to some specific algorithm. In other words, the controller of a crane is usually a human while the controller of a robotic arm is normally a computer program that is able to determine the joint values that provide a desired position or velocity for the end-effector. If we especially consider maritime cranes, compared with robotic arms, they rely on a much more complex model of the environment with which they interact. These kinds of cranes are in fact widely used to handle and transfer objects from large container ships to smaller lighters or to the quays of the harbours. Therefore, their control is always a challenging task, which involves many problems such as load sway, positioning accuracy, wave motion compensation and collision avoidance.

Some of the similarities between robotic arms and cranes can also be extended to robotic hands. Indeed, from a kinematic point of view, a robotic hand consists of one or more kinematic chains fixed on a base. However, robotic hands usually present a higher number of degrees of freedom (DOFs) and consequentially a higher dexterity compared to robotic arms. Nevertheless, several commonalities can be found from a design and control point of views. Particularly, modular robotic hands are studied in this thesis from a design and control point of view.

Emphasising these similarities, the general term of *robotic manipulator* is thereby used to refer to robotic arms, cranes and hands. In this work, efficient design methods for robotic manipulators are initially investigated. Successively, the possibility of developing a flexible control architecture that allows for controlling different manipulators by using a universal input device is outlined. The main challenge of doing this consists of finding a flexible way to map the normally fixed DOFs of the input controller to the variable DOFs of the specific manipulator to be controlled. This process has to be realised regardless of the differences in size, kinematic structure, body morphology, constraints and affordances. Different alternative control algorithms are investigated including effective approaches that do not assume a priori knowledge for the Inverse Kinematic (IK) models. These algorithms derive the kinematic properties from biologically-inspired approaches, machine learning procedures or optimisation methods. In this way, the system is able to automatically learn the kinematic properties of different manipulators. Finally, a methodology for performing experimental activities in the area of maritime cranes and robotic arm control is outlined. By combining the rapid-prototyping approach with the concept of interchangeable interfaces, a simulation and benchmarking framework for advanced control methods of maritime cranes and robotic arms is presented.

From a control point of view, the advantages of releasing such a flexible control system rely on the possibility of controlling different manipulators by using the same framework and on the opportunity of testing different control approaches. Moreover, from a design point of view, rapid-prototyping methods can be applied to fast develop new manipulators and to analyse different properties before making a physical prototype.

Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU), Trondheim, Norway, for partial fulfilment of the requirements for the degree of philosophiae doctor.

This doctoral work has been performed as part of a collaboration between the Department of Engineering Cybernetics, NTNU, and the Department of Maritime Technology and Operations, Aalesund University College (AAUC), Aalesund, Norway, with Professor Kristin Ytterstad Pettersen (Department of Engineering Cybernetics, NTNU) as main supervisor and with co-supervisors Professor Domenico Prattichizzo (Department of Advanced Robotics, Istituto Italiano di Tecnologia (IIT), Genova, Italy) and Professor Houxiang Zhang (Department of Maritime Technology and Operations, AAUC).

Within this thesis, the work concerning robotic hands was partially supported by the European Commission with the Collaborative Project no. 248587, “THE Hand Embodied”, within the FP7-ICT- 2009-4-2-1 program “Cognitive Systems and Robotics” and the Collaborative EU-Project “Hands.dvi” in the context of ECHORD (European Clearing House for Open Robotics Development). In this context, a close cooperation was established with the Department of Advanced Robotics at the Istituto Italiano di Tecnologia and the Department of Information Engineering at the University of Siena, Italy.

Regarding maritime cranes and robot, the work was partly supported by the Research Council of Norway through the Centres of Excellence funding scheme, project number 223254 and the Innovation Programme for Maritime Activities and Offshore Operations, project number 217768. In this context, a close cooperation was established with different partners including Rolls-Royce Marine AS, Huse Engineering and the Offshore Simulation Centre AS.

Acknowledgments

It is hard to overstate my gratitude to my PhD supervisor, Professor Kristin Ytterstad Pettersen. Without her inspirational guidance, her enthusiasm, her encouragements, her unselfish help, I could never achieve these results.

I would like to thank to my other co-supervisors Professor Domenico Prattichizzo and Professor Houxiang Zhang for their inspiring advices and suggestions. I gratefully acknowledge Professor Hans Petter Hildre, Dean of the Department of Maritime Technology and Operations, AAUC, for his support to my research activity and for finding the necessary funding sources that made my PhD work possible.

I am especially thankful to Professor Webjørn Rekdalsbakken from the Department of Engineering and Natural Sciences (AIR), Aalesund University College. He gave me several opportunities to improve both my research as well as my teaching skills.

Moreover, I am grateful to all my colleagues that have contributed immensely to my personal and professional time. These people have been a source of friendships as well as good advice and collaboration.

Particularly, I would like to acknowledge my friends and colleagues Lars Ivar Hatledal, researcher at the Department of Maritime Technology and Operations, AAUC, and Massimiliano Fago, technician at the IMTS S.r.L. Company, Taranto, Italy. We worked together on the implementation of several research projects and I very much appreciated Lars Ivar's and Massimiliano's enthusiasm, intensity and ability to fast-develop well-written computer programs.

Above all, I would like to thank all my students, they are the ones that fill all my efforts and research activities with meaning. Beyond my research, I have been fortunate to obtain a wide-range of teaching experiences at AAUC including the topics of Real-Time Computer Programming, Mechatronics and System Modelling at both bachelor and master levels. Several student projects and thesis have been supervised. I strongly believe that sharing enthusiasm for my research topics is important. My teaching philosophy is to foster a studying environment that promotes enthusiasm, life-long learning, connection to the real world, organised study, student and teacher accountability and confidence building. I try every angle and every strategy to help students learn and succeed.

Last but not least, I would like to thank my family for all their love and encouragement and, especially, my parents Rosario and Concetta who raised me with a love of science and supported me in all my pursuits. My sister Elisa and my brother Riccardo have always been very supportive. Finally, I am grateful to my loving, supportive, encouraging, and patient girlfriend Rūta whose faithful support during the final stages of this PhD is so appreciated. Thank you!

Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

This chapter contains the problem outline. In particular, the motivation for the underlying research work is presented highlighting the research context, the fundamental research questions and themes. Finally, a summary of the main contributions is given.

1.1 Problem Outline

There are some similarities between robotic arms and cranes, particularly regarding their operation and design. Both have various links that are serially or parallelly attached by joints that can be moved with actuators. In both systems, it is possible to move the end-effector to any desired location within the boundaries of the system's workspace and a certain amount of load can be carried. However, traditional cranes are usually cumbersome due to the heavy loads they must move at low speeds. Industrial robots on the other hand are typically smaller, in that they usually move lighter masses at relatively higher speeds. For this reason, cranes are normally actuated using hydraulic valves, while servo motors, pneumatic or servo-pneumatic actuators are used to drive robotic arms.

The most important and fundamental difference between these two kinds of systems is the control method: a human operator usually controls cranes using simple joysticks in joint-by-joint operations, in which each axis corresponds to only one specific actuator, while a computer-operated central controller is typically used to control robotic arms, coordinating the actuators in accordance with a specific algorithm. This means that the controller of a robotic arm is able to determine the joint values necessary for a desired position or velocity of the end-effector.

Therefore, when considering maritime cranes, it is evident that they rely on a much more complex model of their environment when compared to robotic arms. These kinds of cranes are widely used to handle and move objects from large container ships to smaller lighters or harbour quays. As such, controlling cranes is always a difficult task that involves many problems, such as load sway, positioning accuracy, wave motion compensation and collision avoidance.

Some of the similarities that robotic arms and cranes share can also be extended to

robotic hands. When considering kinematics, a robotic hand consists of one or more kinematic chains that are fixed to a base. However, there are usually more degrees of freedom (DOFs) in robotic hands, and consequentially higher dexterity than in robotic arms. Nonetheless, there are several commonalities from the design and control points of view. Particularly, modular robotic hands are studied in this thesis from a design and control point of view.

Emphasising these similarities, *robotic manipulator* is the term generally used to refer to robotic arms, cranes and hands. When planning to control different robotic manipulators, efficient and adaptable design methods are needed. A possible approach consists in developing a flexible control architecture that allows for controlling different manipulators with a universal input device. The biggest challenge in realising such architecture consists of finding a flexible way to map the input controller's normally-fixed DOFs to the specific manipulator's variable DOFs. This process must be realised regardless of differences in size, kinematic structure, body morphology, constraints and affordances. Effective approaches to heave compensation are among some of the alternative control algorithms that are investigated. The methods considered are those that do not rely on a priori knowledge for the Inverse Kinematic (IK) model of the arm. The kinematic properties of these algorithms are derived from biologically-inspired approaches, machine learning procedures or optimisation methods. As a result, the system is able to infer the kinematic properties of different manipulators automatically. Finally, a methodology for performing experimental activities in the area of maritime cranes and control of robotic arms is summarised. A simulation and benchmarking framework is presented for advanced control methods of maritime cranes and robotic arms, which merges the concept of interchangeable interfaces with the rapid-prototyping approach.

Several advantages are evident in the realisation of this kind of flexible control system from both control and design points of view. The same framework can be used for testing different control approaches and controlling different manipulators. Moreover, rapid-prototyping methods can be applied in order to quickly develop new manipulators and analyse different properties before making a physical prototype.

1.2 Related Research Works and Challenges

In this section, a review of the related works is given for the main research areas considered in this thesis: modular robotic hands, maritime cranes and robotic arms. The objective of this section is to provide an insight overview of the current state of the art of the considered fields of study by focusing on the most relevant challenges that this thesis addresses.

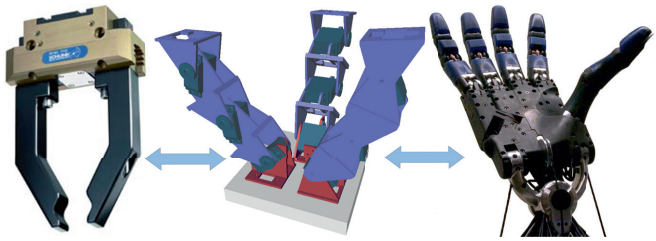


Figure 1.1: Modular grasping allows for finding a trade-off between a simple gripper and more complex human like manipulators.

1.2.1 Related Research Works and Challenges Concerning Modular Robotic Hands

In spite of the great success of bio-robotics in mimicking certain human behavior patterns there is still a large gap between the performance of anthropomorphic robot hands and human hands. Human hands are capable of grasping an astounding variety of objects of different shapes, textures, weights and spatial orientations. Building a robotic hand with sufficient dexterity and multi-degrees of freedom has become one of the most attractive steps in order for a robot to fully mimic the movement of the human hand. However, development of such hands is challenging because it is required to fit large number of degrees of freedom.

A possible solution consists in limiting the device to the minimum number of degrees of freedom necessary to accomplish the desired task. In [1], the kinematic behavior of the human hand was analysed in order to obtain simplified human hand models with minimum and optimal degrees of freedom, and thus achieving an efficient manipulation task. The main disadvantage of this approach is that such simplified robotic hands are usually difficult to adapt to different grasping operations or to the grasping of objects with dissimilar size.

Another promising approach to get such flexibility is to use a modular approach [2, 3]. The modular approach allows using only the necessary number of DOFs to accomplish the grasp. In this way it is possible to find a trade-off between a simple gripper and more complex human like manipulators. The idea is shown in Figure 1.1. Moreover, great advantages are obtained in versatility since the robotic hand can be disassembled and reassembled to form new morphologies that are suitable for new tasks. Modularity offers also robustness considering that robot parts are interchangeable [4]. The production cost can also be considerably cut by building a specialised device capable of grasping objects by using only the number of actuators and DOFs required. Besides, the weight of the manipulator would be minimized to the bare necessities.

However, there are several challenges to be addressed including design issues, prototyping problems and control questions. These challenges are discussed in the following.

1.2.1.1 Design Challenges

In order to fully exploit the flexibility offered by the modular concept, an efficient design algorithm, which allows for finding effective modular configurations according to the task to be performed, is necessary. In literature, some initial studies are related to the self-reconfigurable robots. A generally applicable task-related objective function which evaluates a modular robot assembly configuration for a given task was presented in [5]. The authors used an optimization method based on GA. In [6], a cellular robot capable of adapting its shape and functions to changing environments and demands by rearranging its mutual mechanical connection was presented. In [7], the authors proposed an algorithm for grasping objects with a self-reconfigurable system. Although the idea that a modular gripper can handle objects of unknown shape and size was pointed out, the work reported preliminary results still far from a real implementation. A generalised distributed consensus framework for self-adaptation tasks in modular robotics was introduced in [8]. It was shown by the authors that a variety of modular robotic systems and tasks can be formulated within such a framework. The authors presented three main contributions: an adaptive column that can respond to external forces, a modular tetrahedral robot that can move towards a light source, and a modular gripper that can wrap around fragile objects. The decentralized control used in their work is based on the sharing of information about pressure given by sensors included in each module. This solution is not applicable when, for instance, fingertip manipulation is required. Furthermore, investigation on grasp stability is missed by the authors.

To the best of our knowledge, few works investigate the possibility of developing an efficient design algorithm that allows for finding effective modular configurations to get efficient grasps of given objects.

1.2.1.2 Prototyping Challenges

From a design point of view, rapid-prototyping can be beneficial when developing modular manipulators with different configurations. Development time can be significantly reduced, the main grasp properties can be analysed and the quality can be assessed. Therefore, rapid-prototyping is a necessary step to validate the design before making a physical prototype.

Robotic systems have been used as part of a rapid prototyping process [9, 10]. However, the application of rapid-prototyping in robot design has been very limited, especially concerning robotic hands. In [11], prototypes of mechanical joints were fabricated experimentally and then used to build the articulated structure of one 4-DOFs finger on a five-fingered robotic hand. In this case, the joints showed good smoothness and evenness in flat vertical and horizontal surfaces. Indeed, joint compliance can enable successful robot grasping despite uncertainties in target object location. Compliance also enhances manipulator robustness by minimising contact forces in the event of unintended contacts or impacts. In [12], the design, fabrication, and evaluation of a novel compli-

ant robotic grasper constructed by using polymer-based shape deposition manufacturing (SDM) was presented. Afterwards, in [13], the same manifacturing technology was used to build a four-fingered, underactuacted manipulator. This gripper was designed with rapid-prototyping methods to provide performance adequate for general-purpose experimentation, while requiring only off the-shelf components and minimal machining. Simple 3-D printed components were used to make the hand compact and lightweight.

However, most of these previous works mainly focus on the mechanical construction process, while hardware, control and software prototyping are often neglected in the prototyping design. To the best of our knowledge, an integrated mechanical, hardware and software rapid-prototyping framework for designing and testing different configurations of modular grasping manipulators is still missing.

1.2.1.3 Control Challenges

Modular hands present a great potential in terms of versatility, robustness and low cost. However, programming such robots for specific grasping tasks can be challenging. In this regard, a software architecture that fully exploits the concept of modularity is required. Over the past few years, the possibility of creating such a kind of software framework for robotic hands has been investigated by several research groups. For instance, in [14], a control system architecture for the DLR Hand II of the German Aerospace Center was presented. A multilevel, modular structure of the whole hand system was also introduced in the hand's software architecture. The developed concept of modular levels was designed mainly to perform multiple different tasks on a higher abstraction level. Another notable example of modular control architecture was presented in [15], where the application of a virtual decomposition control approach to modular robot manipulators is discussed. A high-speed data-bus with a data rate of 100 Mbps is used for necessary information exchange among the modules. The dynamics-based control is fully handled by the local embedded controllers, whereas the host computer handles the kinematics related computation. The stability of the entire robot is rigorously guaranteed.

However, most of these previous works mainly focus on building a framework that often applies to a specific modular system, while the idea of simplicity, the objective of practicality and the concept of rapid-prototyping are often neglected in the design of the control architecture. In this prospective, the extreme versatility of the modular grasping requires a completely new paradigm concerning both hardware and software design. To the best of our knowledge, a flexible software architecture that takes advantages of the underlying modular hardware with simplicity in mind is still missing.

Another open question concerns the choice of a flexible input device to control the designed modular hands. One of the most fascinating and promising approach consists of using using a brain-computer interface (BCI), several researchers are recently trying to concentrate their efforts and investigations on this topic and, especially, on the adoption of technologies based on the use of electroencephalography (EEG). This choice opens up to a variety of possible applications including the development of modular prototypes for

mind-controlled prosthetic hands. In particular, with the latest advances in the technology that allows for monitoring and processing the human electroencephalographic signal, increasingly promising and non-invasive approaches are attracting more attention. Nonetheless, few studies have demonstrated practical BCI control of robotic modular manipulators. Most of the previous works focus on the control of prosthetic devices that do not exhibit a modular design in terms of both hardware and software. For instance, in [16], an EEG-based motor imagery BCI was presented to control the movements of a prosthetic hand. The hand was instrumented with force and angle sensors to provide haptic feedback and local machine control. Using this system, subjects demonstrated the ability to control the prosthesis's grasping force with accuracy. In [17], the design of a wearable mind-controlled prosthetic hand, based on the use of a commercial EEG headset, was presented. Decision-making methods, intelligent control of the prosthetic hand and the man-machine coordinated approaches were studied. The hand was equipped with pressure sensor arrays to imitate the touch and slip feelings. Besides, an accelerometer sensor and an angular velocity sensor were used to acquire the feedback of the prosthesis's position and orientation.

However, from a computation point of view, these previous works involve quite demanding control algorithms, which can be hardly distributed on a modular architecture. In addition, different sensors are required to achieve the control objectives. These characteristics do not easily match with the principles of minimalism, simplicity and low-cost that are at the base of modular robotics.

1.2.2 Related Research Works and Challenges Concerning Maritime Cranes and Robotic Arms

Pertaining to maritime cranes and robotic arms there are several common challenges that can be identified including low control flexibility and non-standardisation issues, control problems and performance evaluation. These challenges are discussed in the following.

1.2.2.1 Low Control Flexibility and Non-Standardisation

Maritime Cranes. In the maritime industry, the last few decades have seen a growing interest in developing new technologies for controlling modern vessels and related maritime equipment to perform increasingly demanding marine operations. One of the biggest challenges concerns the operation of maritime cranes. Cranes are widely used to handle and transfer objects from large container ships to smaller lighters or to the quays of the harbours. The control of maritime cranes is always a challenging task, which involves many problems such as load sway, positioning accuracy [18], wave motion compensation, collision avoidance [19] and manipulation security [20]. Moreover, traditional on-board maritime cranes, which are relatively big, heavy and stiff, rely on complex kinematic models of their system as well as an equally complex model of the environment with which they interact.

Figure 1.2: The plethora of the currently used input devices for controlling maritime cranes.

Even though the operating environment can be very challenging, it is still quite common to use relatively simple control interfaces to perform offshore crane operations [21]. In most cases, the operator has to handle an array of levers, throttles or buttons to operate the crane joint by joint. Moreover, each input device can normally control only one specific crane model. The plethora of the currently used input devices for controlling maritime cranes is shown in Figure 1.2. When considering working efficiency and safety, this kind of control is extremely difficult to manage and extensive experience with high control skill levels is required of the operators [22]. Therefore, low control flexibility and non-standardisation are two crucial issues of the current maritime crane control architecture that need to be overcome.

In existing literature, not much work has been done to overcome the low control flexibility and non-standardisation problems for the current maritime crane control architecture. A tele-robotic controlled handling system operated by an intuitive controller was proposed in [23]. The system combines tele-robotic control with an off the shelf marine crane and a custom designed end-effector. The controller is designed such that movement in a certain direction corresponds to an identical movement of the crane. However, the human operator is suppose to act as the motion compensator and to control the end-effector. To improve the operator's experience, an intuitive system that can provide suggestive information for the crane operator via haptic feedback was developed in [24]. Operational support was provided for the operator's input device via the application of the impedance control and gravity compensation. Restrictions on the velocity of the crane were imposed. This control system allows the operator to safely transfer a load to arbitrary positions without colliding with obstacles.

However, most of these previous studies only concern the control of a specific crane/arm.

Figure 1.3: The plethora of the currently used robot controllers from different robot manufacturers.

Very little work has been done regarding the possibility of controlling different arms by using the same input device. To the best of our knowledge, a universal input device that allows for controlling of different models of maritime cranes by using the same universal input device regardless of their differences in size, kinematic structure, degrees of freedom, body morphology, constraints and affordances, has not been released yet.

Robotic Arms. From a similar prospective, a non-standardisation issue also affects robotics arms. In particular, there is a lack of a common and standard control interface that would allow for controlling different manipulators. The plethora of the currently used robot controllers from different robot manufacturers is shown in Figure 1.3. Industries that employ robots in a wide variety of applications are the main customers for robot manufacturers. The manipulator market for research applications, on the other hand, is simply too small for the robot manufacturing industry to develop models specifically for such use. While the hardware and mechanical requirements of developed robots are often similar for both industry and research, scientific software requirements are quite different and even contradictory in many aspects [25], [26]. The goal of scientists is to try to gain as much control over the robot as possible, whereas industries seek safe and easy operational interfaces. In particular, although software interfaces that are appropriate for industrial use are available, it is difficult to find interfaces that are applicable for research purposes. The disclosure of the internal control architecture is also very hard to come by. Many manufacturers are unwilling to publish internal details regarding system architecture due to the high levels of competition in the robot market. Consequently, it is not possible to fully exploit many robotic platforms in a scientific context. Only a small number of industrial manipulators with an open control interface has been released. The

few standard interfaces are usually limited to a small number of specific models.

Focusing exclusively on *Kuka* robots [27], the *Kuka Robot Language* (KRL) is the standard programming language [28]. It is a text based language that offers data type declaration, specification of simple motions, and interaction with tools and sensors via I/O operations. It is only possible to run KRL programs on the *Kuka Robot Controller* (KRC), where program execution is done in accordance with real-time constraints. While the KRL offers an interface that is easy to use in industrial applications, it is quite limited for research purposes. In particular, the KRL is tailored to the underlying controller and consequently, only a fixed, controller-specific set of instructions is offered [29]. Advanced mathematical tools such as matrix operations, optimisation or filtering methods are not supported, thus making the implementation of novel control approaches very difficult. There is no native way to include third party libraries and as such, extending the KRL to include new instructions and functionalities is an arduous task. Moreover, it is not possible to directly use external input devices. The standard workaround for partially expanding the robot's capabilities is to use supplementary software packages provided by *Kuka*. Some examples of such packages are the *Kuka.RobotSensorInterface* [30], which allows the manipulator motion or program execution to be influenced by sensor data, and the *Kuka.Ethernet KRL XML* [30], a module that allows the connection of the robot controller with up to nine external systems (e.g. sensors). However, several drawbacks accompany these supplementary software packages: I/O is limited, a narrow set of functions is present and major capital investments are often required to actually purchase these packages from *Kuka*.

The possibility of creating a software interface to *Kuka* robots has been investigated by several research groups. An open-source real-time control software for the *Kuka lightweight* robot, *OpenKC*, was presented in [25]. This software makes it possible to externally trigger and control all of the features of the *Kuka lightweight* (LBR) manipulator. This is done by using a simple set of routines that can easily be integrated into existing software. As a result, developers of robot applications have an edge in finding solutions for a variety of different software scenarios. In particular, force and torque readings as well as different modes of operation can be remotely read and parametrised. However, this software interface is restricted to a specific model of *Kuka* robots, the *Kuka lightweight* manipulator, and use of the *Kuka.RobotSensorInterface* package is required. Another interface that is currently available for the *Kuka lightweight* robots is the *Fast Research Interface* (FRI) [26]. The FRI provides direct low-level real-time access to the KRC at high rates of up to 1 kHz. On the other hand, all features, like teaching, motion script features, fieldbus I/O and safety are provided. The FRI is based on the *KR C*2. Without much installation efforts, access to different controller interfaces of the *Kuka* system is provided including joint position control, cartesian impedance control, and joint position control. However, also this software interface is restricted to a specific model of *Kuka* robots, the *Kuka lightweight* manipulator. No support for the standard *Kuka* industrial robots is provided.

Later on, the *Kuka Control Toolbox* (KCT), a collection of *MATLAB* functions for motion

Table 1.1: Currently available interfaces for *Kuka* robots

Interface	Support to *Kuka* LBR	Support to *Kuka* industrial robots	External packages required
OpenKC	Yes	No	Yes
FRI	Yes	No	No
KCT	No	Yes (only small and low-payload)	Yes
Robotics APIs	Yes	Yes (safety limitations)	No
ROS	Yes	No	No
KUKA Sunrise.Connectivity	Yes	No	Yes

control of *Kuka* industrial robots, was introduced in [31] to offer an intuitive and high-level programming interface for the user. This toolbox is compatible with all small and low-payload *Kuka* robots that have six degrees of freedom (DOFs). The KCT runs on a remote computer connected to the KRC via TCP/IP. A multi-thread server runs on the KRC and communicates via *Kuka.Ethernet KRL XML* with a client whose job is to manage the information exchange with the manipulator. High transmission rates are guaranteed by this communication set-up, thus enabling real-time control applications. Nonetheless, as in the previous work, this approach is still tailored to the underlying controller and requires the use of the *Kuka.Ethernet KRL XML* package.

A different approach has been tried by other researchers, aimed at the disclosure of the *Kuka* industrial manipulator internal control architecture. For instance, the reverse engineering of the KRL was investigated in [29] and a set of Java-based *Robotics APIs* was presented for programming industrial robots on top of a general-purpose language. The *Robotics APIs* implement robot commands like motions and access to I/O calls. It was shown that KRL can be bridged by batch-executing motions, under the assumption that executing control flow and calculation statements takes only a small amount of time compared to the time it takes the robot to complete a motion command. However, some safety limitations are inherently present in the *Robotics APIs* set because it is the result of a reverse engineering approach and therefore does not include a way of specifying complex triggers in contrast to the KRL.

In the last few years, the Robot Operating System (ROS) [32], an open-source software toolbox for robotic development, has become more and more popular among the research community. The primary goal of ROS is to provide a common platform to make the construction of capable robotic applications quicker and easier. Some of the features it provides include hardware abstraction, device drivers, message-passing and package management. ROS provides support for different industrial robots including vendors like ABB, Adept, Fanuc, Motoman and Universal Robots. Extensive research work has also gone into creating ROS packages for communicating with the *Kuka lightweight* robots but no support is provided for the *Kuka* standard industrial robots yet. One of the main reasons for this lack is the non-disclosure of the KUKA Robot Controller (KRC) internal

architecture which currently makes it impossible to directly interact with the robot to be controlled.

Recently, *Kuka* has shown more interest in the research and education market. In particular, the *KUKA Sunrise.Connectivity* has been recently developed for *Kuka lightweight* robots. This software provides a collection of interfaces for influencing robot motion at various process control levels. Third-party software can be easily integrated into the user-specific application using the popular standard programming language Java. Along with the quick update of the target position directly from the robot application, it is also possible to access the robot controller from external computers in hard real-time mode. However, even in this last case, the main limitation is that this software is restricted to the *Kuka lightweight* manipulators.

To provide a more clear overview of the currently available interfaces for *Kuka* robots, a table of comparison of all the reviewed related works is shown in Table 1.1.

To the best of our knowledge, a cross-platform communication interface that works with all *Kuka* industrial robots without requiring any external packages has not been released yet.

1.2.2.2 Control Challenges

Control Methods for Maritime Cranes or Robotic Arms. Typically, different input devices are used to control specific manipulators. Moreover, a custom control algorithm is usually developed for each specific manipulator to be controlled as shown in Figure 1.4. In order to design a control algorithm for a crane or for a robotic arm, the kinematic properties of the system needs to be found. One approach is to derive the inverse kinematics (IK) model to be controlled. Commonly, this approach enables researchers to either introduce analytical methods, which offer exact solutions for simple kinematic chains, or propose solutions based on numerical methods. However, when considering arms with redundant degrees of freedom, the inverse kinematics can have multiple solutions; therefore, singularity problems could arise. In addition, this method is not very flexible, especially when planning to control different arms using a universal input device because several IK models are needed: one for each arm or crane to be controlled. An alternative solution to the problem might consist of using methods that derives the kinematic properties by applying an optimisation approach. In this way, the system would be able to automatically obtain the kinematic properties of different arms, and new models could also easily be added.

During the last few years, there has been increasing interest regarding research on learning algorithms and many efforts have been made to understand how to apply this technology to various control problems. In particular, several Genetic Algorithm (GA) models [33] have been developed by applying biologically-inspired control mechanisms to robot control tasks. For instance, an approach for a robot inverse velocity solution using GA was proposed in [34]. The authors used the principle of robot motion propagation from link to link to find the robot's recursive velocity formula, which then was used to determine

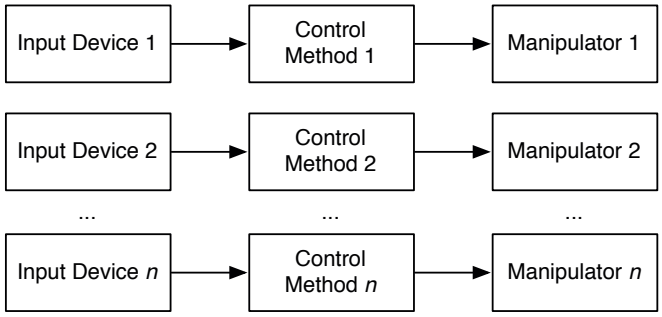


Figure 1.4: Typically, different input devices are used to control specific manipulators. Moreover, a custom control algorithm is usually developed for each specific manipulator to be controlled.

the fitness function. In [35], a GA approach was used to solve for multiple solutions of inverse kinematics using adaptive niching and clustering. The niching method was used to modify the GA fitness value to encourage convergence around multiple solutions in the search space. The authors concluded that the proposed algorithm could be generalised to solve the IK problem of a robot with unknown DOFs and configuration, and that the method worked with good precision and speed.

Similarly, several Artificial Neural Network (ANN) models [33] have been developed to tackle the same challenge. Especially, in order to deal with complex robotic systems and with the related non-linear problems that arise when considering sophisticated types of actuators, several ANN models have been developed. In [36], a Lagrangian neural network was presented for the IK computation of redundant manipulators based on the Euclidean norm of the joint velocities. This was developed at first to show its feasibility. Next, in the same work a primal-dual neural network for minimum infinity norm kinematic control was presented. To reduce the model complexity and increase the computational efficiency, a dual neural network was finally introduced with the advantages of simple architecture and exponential convergence. However, the simulation results are based only on a specific industrial robot, the *PA10* robot manipulator. In [37], a spiking neural network architecture was developed that autonomously learns to control a 4 DOFs robotic arm after an initial period of *motor babbling* (motor babbling can be observed in babies, where a repetitive action-perception cycle generates associative information between the various representations). The spiking neurons have been simulated according to Izhikevich's model [38], which exhibits biologically realistic behaviour and yet is computationally efficient. These works demonstrate that ANNs can be also used to model complex relationships between inputs and outputs.

Analogously, different models based on Particle Swarm Optimization (PSO) [39] have been presented to deal with robot control tasks. For instance, a method based on PSO was used to search the global time-optimal trajectory for a space manipulator in [40]. For this formulation, some inter-knots of joint trajectory are defined as optimal parameters. These

inter-knot parameters mainly include joint angle and joint angular velocities. Finally, an illustrative example was presented to verify that the developed PSO-based time-optimal trajectory planning method has satisfactory performance and real significance in engineering. In [41], a study originated on swarm intelligence was presented. Three case studies on one link arm, a *SCARA* robot and a electrohydraulic robot were presented on both simulation and experimental base to show effectiveness and simplicity of application.

On the other hand, most of these previous intelligent systems are only able to infer the control of a specific crane/arm. To date, it is not common to use a universal input device to control various cranes/arms with different kinematics. Moreover, most of these works require the same DOFs for both the input device and the model to be controlled.

Additional Challenges for Offshore Cranes. Unlike cranes mounted on fixed bases, offshore crane operations are significantly influenced by the ship motions resulting from currents and waves. The dynamic forces generated from the heave motion of the vessel and the sway movements of the load pendulation have significant effects on the crane operations. Operating in such a challenging scenario is very demanding. Advanced control methods are needed in order to compensate for the wave impact and to guarantee efficiency and safety.

Numerous research efforts and investigations have been done to help reduce the risk in offshore crane operations. Focusing exclusively on the heave compensation problem, two different approaches have been extensively investigated. The first technique, Passive Heave Compensation (PHC) [42], was the first to be proposed and is the simplest of these two approaches. A PHC system can simply be modeled as a spring damper system by means of hydraulic cylinders and compressors. The second method, Active Heave Compensation (AHC) [43], differs from PHC by having controlled actuators that actively try to compensate for the heave movements. To monitor the ship movements, commercial offshore cranes usually adopt some motion detection units, e.g. Inertial Measurement Unit (IMU) and Motion Reference Unit (MRU). Then, according to this data input, a control system calculates how the actuators have to react to the movements. The actuators can be electric or hydraulic winch systems or hydraulic cylinders.

Due to the challenging crane operational scenario in real applications, several studies have been performed by using a computer-simulated environment. For instance, a heave compensation system based on heave motion prediction and an inversion based control strategy was proposed in [44]. In particular, a combination of a trajectory tracking disturbance decoupling controller and a prediction algorithm was presented and evaluated with simulation and measurement results. Recently, our research group adopted a computer-simulated environment to develop an effective heave compensation and anti-sway control approach for offshore crane operations, which is based on robotic arm kinematics and energy dissipation principles [45]. Unlike common operator-based joint-by-joint control procedures, this automated method is more flexible, allowing for more intuitive crane operations and more accurate positioning of the hoisted load. In particular, a unique feature of this approach is that the two control functions of heave compensation and anti-sway

are transparently combined and simulated in an integrated modelling environment.

However, a simulation approach is always limited when compared to a realistic experimental setup. For this reason, other researchers explored the possibility of replicating a laboratory experimental arrangement for performing these kinds of studies. For example, an inverse kinematic control strategy that uses the actuation capability of two cranes (hoist lengths and boom angles) to keep its load fixed in inertial space regardless of the motion of the ship on which the cranes are mounted was presented in [46]. Unique crane commands are computed using a minimum norm solution and a dynamic simulation can be achieved. A final verification of the system was performed using two cranes mounted on a motion controlled platform. In [47], a method for reducing the cargo pendulation was proposed based on the control of the slew and luff angles of the crane boom. The effectiveness of the method was demonstrated in a fully nonlinear three-dimensional computer simulation and in an experiment with a scale model of the considered crane mounted on a platform moving with three DOFs.

Nonetheless, most of these previous works focus on the development and validation process of a specific control method for very distinct crane models. To the best of our knowledge, a general framework that allows for both reproducing in a laboratory setup the same challenging operation scenario as that of maneuvering offshore cranes and for testing different models and control approaches has not been released yet.

1.2.2.3 Performance Evaluation Challenges

Benchmarking as a means of objective comparison and competition among researchers is and has always been of great interest in robotics. The practice of comparative evaluation of different algorithms becomes increasing useful when applied to a well-defined system in a specific domain, as opposed to the field robotics in general. For this reason, efforts are being made to establish standard benchmarks for several robotic fields, including grasping manipulators, mobile robots, human-robot interaction, and robotic arms. In this context, networks and societies, such as EURON or IEEE, play an important role in working on defining standard benchmarking methodologies [48].

Usually, when trying to benchmark several robotic systems, a standard reference environment, reference tasks, and related performance metrics are to be defined. However, it is difficult to define a benchmark that is commonly accepted by the community mainly because of divergent viewpoints on a problem from different research groups. In addition, the risk of fostering the development of specialized solutions for an abstracted, standardized setting exists. When considering robotic arms, a quite common approach to avoid these problems consists in organising scientific competitions and contests where benchmarks are usually discussed and then accepted by all the participants. Different famous competitions are known for complete integrated robotic systems, such as the DARPA Grand Challenges [49], or RoboCup@Home [50] – a competitive scenario for service robots. Unfortunately, participation in such big events is usually limited to a few selected groups, often simply because of limited resources and the lack of necessary hardware. In

addition, this same approach is difficult to apply to maritime cranes, mainly because of their size and complex operation scenarios.

To overcome these limitations, another possible way to compare different control methods relies on the idea of using a virtual environment that closely simulates the desired systems and allows for replicating a set of reference tasks and related performance metrics. For instance, concerning the field of grasping manipulation, a software environment for the comparative evaluation of algorithms for grasping and dexterous manipulation is presented in [51]. This tool allows the reproduction of well-defined experiments in real-life scenarios in every laboratory and, hence, it provides benchmarks that pave the way for objective comparison and competition in the field of grasping. Considering the robotic planning domain, a benchmark tool for multi-robot simulation is presented in [52]. In the field of autonomous mobile robots, a unified benchmark framework for evaluating and comparing motion algorithms for autonomous mobile robots and vehicles is introduced in [53]. Focusing exclusively on robotic arms, a control engineering benchmark problem with industrial relevance is presented in [54]. The process is a simulation model of a nonlinear four-mass system, which should be controlled by a discrete-time controller that optimizes performance for given robustness requirements. Nonetheless, the control problem concerns only the so-called regulator problem. In [55], a benchmark problem for robust feedback control of a manipulator is introduced. The system to be controlled is an uncertain nonlinear two-link manipulator. The control problem concerns only disturbance rejection. The proposed model is validated by experiments on a real industrial manipulator. However, most of these previous works mainly consider only a specific manipulator model, and it is not possible to dynamically exchange models or control methods.

As opposed to the field of robotic arms, where at least a few benchmark suites and methods for estimating the efficiency of the considered control approach already exist, in the field of maritime cranes there is a lack of a universally recognised benchmarking method for assessing the system performance. This is the main reason why it currently is extremely difficult not only to compare results of different control approaches, but also to assess the quality of the research presented by the authors. To the best of our knowledge, no standard benchmarking tools are currently available in this field – neither in the form of competitions, nor as routines to run in a simulation environment.

1.3 Scope of the Thesis

The following subjects form the scope of the research work that is presented in this thesis.

1.3.1 Design Methods and Control Architecture

One of the main objectives of this thesis is to tackle the challenge of developing efficient design methods for robotic manipulators with dissimilar configurations. The possibility

of realising a flexible control architecture for different robotic manipulators is also investigated. In particular, the possibility of developing a framework that allows for controlling different manipulators independently from their specific structure is investigated. This framework should also allows for transparently selecting different control approaches. In other terms, the possibility of creating a *middleware* solution that allows for switching between different robots and between alternative control algorithms by focusing focus only on the task to be accomplished is explored.

1.3.2 Alternative Control Algorithms

Another objective of this thesis is to tackle the low control flexibility and non-standardisation issues that currently effect the field of robotic manipulators. To do so, the possibility of developing alternative control algorithms that are able to scale and handle different manipulator configurations, from simple ones with few DOFs to the most complex with a considerable number of DOFs, is considered.

The objective is to investigate control methods that do not assume a priori knowledge for the IK model of the robotic manipulator to be controlled. In this prospective, the possibility of deriving the kinematic properties from biologically-inspired approaches or optimisation methods is investigated.

1.3.3 System Integration

Different integration challenges are considered. In particular, the integration between virtual and real prototypes is investigated. The objective is to create a system integration that allows for establishing a real-time one-to-one correspondence between virtual and physical prototypes.

The system integration between a flexible control architecture with a simulation environment specifically designed for particular application scenarios is also considered. With this kind of integration, the objective is to provide specific support to researchers according to different fields of application.

The system integration and control of real industrial robotic arms is also investigated. In particular, the objective is to develop a standard control interface that allows researchers to use different input devices, sensors and to develop alternative control methods. This system integration is relevant for both research and industrial applications.

1.3.4 Benchmarking Different Control Methods

The challenging problem of assessing the performance of different control methods is addressed. In particular, the objective is to design a set of routine tests, different cost

functions, and metrics. The purpose is to build a benchmark suite for robotic manipulators according to different application scenarios.

The intent is to transparently integrate the benchmark suite with the proposed flexible control architecture. This integration can give researchers an important tool for objective comparison and competition.

1.4 Contributions of the Thesis

The title of the thesis is "Alternative and Flexible Control Methods for Robotic Manipulators". The term robotic manipulator is adopted to emphasise the similarities between robotic hands from one side and maritime cranes and robotic arms to another side. As the title describes, the main objective of this thesis is to contribute to further developments concerning robotic manipulators. Following this main goal, several contributions to the manipulator design, system architecture, control methods and benchmarking are presented and described in details below.

Chapter 2

Topic: The possibility of developing efficient design methods and the challenge of developing a flexible control architecture for different robotic manipulators is considered.

Contributions: concerning modular robotic hands, the first contribution of this chapter is to define a generalised modular model for robotic grasping. Based on this model, a simulation-based algorithm that allows for finding effective modular configurations to get efficient grasps of given objects is presented. This method makes it possible to design modular hands that use only the necessary number of DOFs to accomplish a given grasping task. Therefore, the contribution of this method is relevant from both a scientific as well as an application point of view. Nonetheless, the proposed design method is originally developed in a simulated environment and consequently it only allows for designing and testing virtual modular manipulators. To overcome this limitation, *ModGrasp*, an open-source virtual and physical rapid-prototyping framework is successively proposed. This framework represents a significant contribution because it allows for establishing a real-time one-to-one correspondence between virtual and physical prototypes.

With regard to the study of maritime cranes and robotic arms, a generalised manipulator model is presented. Based on this model, a flexible control architecture for different manipulator models is introduced as one of the main contributions of this chapter. This contribution is particularly relevant because it represents the base for the research of flexible and transparent control methods.

Chapter 3

Topic: the possibility of developing alternative control algorithms that are able to scale and handle different robotic manipulator configurations is tackled.

Contributions: pertaining to modular robotic hands, a novel control method is presented in this chapter based on a biologically-inspired idea. In particular, this method adopts the use of human synergies to control the models, as described in [56]. The adoption of this method is relevant for this thesis because it allows to transparently control different modular hand configurations independently from their number of DOFs. To show the potential of this approach, this method is implemented with *ModGrasp*, the open-source virtual and physical rapid-prototyping framework presented in Chapter 2. As a case study, a mind-controlled, low-cost modular manipulator is also proposed.

Correspondingly, the possibility of developing alternative control methods concerning the field of maritime cranes and robotic arms is investigated. The contribution to this topic consists of presenting two alternative control methods that aim to transparently control different manipulators and configurations. The first method is based on the use of Genetic Algorithms (GAs) [33]. The second method involves the use of Particle Swarm Optimisation (PSO) [39]. Both the proposed control methods are relevant for this thesis because they do not assume a priori knowledge for the IK models. These algorithms derive the kinematic properties from optimisation methods. The proposed methods are implemented based on the flexible control architecture for maritime cranes and robot presented in Chapter 2.

Chapter 4

Topic: different integration challenges are considered including the integration between virtual and real robotic manipulators, the integration between a flexible control architecture and a simulation environments specifically designed for particular application scenarios, and the integration and control of real industrial robotic arms.

Contributions: referring to modular robotic hands, the interconnection between virtual and real prototypes from an integration point of view is analysed. In this regard, implementation details about the integration of real modular manipulators with the rapid-prototyping framework of *ModGrasp*, which is presented in Chapter 2, are provided. This system integration is a relevant contribution for the thesis because it allows for conducting experiments with both virtual and real modular prototypes in a transparent fashion.

Correspondingly, the system integration challenge is addressed in the field of maritime cranes and robotic arms. Particularly, the flexible control architecture for maritime cranes and robots presented in Chapter 2 is integrated with a simulation environment specifically designed for offshore applications. The considered simulation environment is the Crane Simulator from the Offshore Simulation Centre AS (OSC) [57]. This system integration is a relevant contribution because it establishes the base for the research of alternative

control algorithms, which can be efficiently tested in a realistic maritime simulation environment.

Next, the system integration and control of real industrial robotic arms is studied. Restricting the focus to *Kuka* industrial robots, *JOpenShowVar*, an open-source cross-platform communication interface to *Kuka* industrial robots that allows for reading and writing variables and data structures of the controlled manipulators is presented. *JOpenShowVar* is an important contribution because it opens up to a variety of possible applications making it possible to use different input devices, sensors and to develop alternative control methods.

Finally, the cranes and robots control system, presented in Chapter 2, is integrated with a physical motion platform, which simulate the wave effects. This system integration is important because it gives researchers the possibility of testing alternative control algorithms for maritime cranes and robotic arms in a realistic and safe laboratory setup.

Chapter 5

Topic: the challenging problem of assessing the performance of different control methods for robotic manipulators is considered.

Contributions: with regard to modular robotic hands, the computation of grasp quality indices is studied in order to assess different grasping methods. In particular, the grasp measuring index adopted in the design method for modular grasping hands, which was presented in Chapter 2, is described. This grasp measuring criteria was previously introduced in [58]. The motivation for this choice are provided.

In a similar way, the issue of benchmarking different control methods is successively investigated in the field of maritime cranes. In particular, a benchmark suite for advanced control methods of maritime cranes is presented as one of the main contribution of this chapter. This suite is transparently integrated with the control architecture presented in Chapter 2 so that it is possible to model different manipulator models, all the corresponding hydraulic systems, various vessels, and the surrounding environment for visualisation. Different control methods can be easily implemented and tested. A set of routine tests, different cost functions, and metrics are provided. This benchmark suite is important because it allows for comparing different control methods independently from the specific crane model to be controlled.

1.5 Publications

The materials presented in this thesis are based on several conference and journal papers which are listed below with full bibliography.

- **F. Sanfilippo**, G. Salvietti, H. Zhang, H. P. Hildre and D. Prattichizzo, **"Efficient Modular Grasping: an Iterative Approach"**, in Proc. of the IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob), Rome, Italy, 2012, pp. 1281-1286.

  This paper introduces a new modular approach to robotic grasping that allows for finding a trade-off between a simple gripper and more complex human like manipulators. The modular approach to robotic grasping aims to understand human grasping behavior in order to replicate grasping and skilled in-hand movements with an artificial hand using simple, robust, and flexible modules. In this work, the design of modular grasping devices capable of adapting to different requirements and situations is investigated. A novel algorithm that determines effective modular configurations to get efficient grasps of given objects is presented. The resulting modular configurations are able to perform effective grasps that a human would consider "stable". Related simulations were carried out to validate the efficiency of the algorithm. Preliminary results show the versatility of the modular approach in designing grippers.

- **F. Sanfilippo**, H. Zhang, K. Y. Pettersen, G. Salvietti and D. Prattichizzo, **"ModGrasp: an Open-Source Rapid-Prototyping Framework for Designing Low-Cost Sensorised Modular Hands"**, in Proc. of the IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob), S*ã*o Paulo, Brazil, 2014, pp. 951-957. **Finalist candidate as IEEE BioRob2014 Best Student Paper Award**.

  This paper introduces *ModGrasp*, an open-source virtual and physical rapid-prototyping framework that allows for the design, simulation and control of low-cost sensorised modular hands. By combining the rapid-prototyping approach with the modular concept, different manipulator configurations can be modelled. A real-time one-to-one correspondence between virtual and physical prototypes is established. Different control algorithms can be implemented for the models.

  By using a low-cost sensing approach, functions for torque sensing at the joint level, sensitive collision detection and joint compliant control are possible. A 3-D visualization environment provides the user with an intuitive visual feedback.

  As a case study, a three-fingered modular manipulator is presented. Related simulations are carried out to validate efficiency and flexibility of the proposed rapid-prototyping framework.

- **F. Sanfilippo**, H. Zhang and K. Y. Pettersen, **"The New Architecture of *ModGrasp* for Mind-Controlled Low-Cost Sensorised Modular Hands"**, in Proc. of the IEEE International Conference on Industrial Technology (ICIT), Seville, Spain,

2015, pp. 524-529. **Student Scholarship Award**.

In this work, the *ModGrasp* communication pattern is improved, becoming more modular, reliable and robust. In the previous version of the framework, each finger of the prototype was controlled by a separate controller board. In this work, each module, or finger link, is independent, being controlled by a self-reliant slave controller board. In addition, a newly redesigned multi-threading and multi-level software architecture with a hierarchical logical organisation is presented. In this regard, a new programming paradigm is delineated.

The new architecture opens up to a variety of possible applications. As a case study, a mind-controlled, low-cost modular manipulator is presented. In detail, the user's levels of attention and meditation are monitored by using an electroencephalography (EEG) headset, the *NeuroSky MindWave*. These levels are used as inputs to control the hand. Since the manipulator features 11 DOFs, a synergistic control approach is chosen to map inputs with outputs with such a different dimensionality. Related simulations and experimental results are carried out.

- **F. Sanfilippo**, L. I. Hatledal, H. G. Schaathun, K. Y. Pettersen and H. Zhang, **"A Universal Control Architecture for Maritime Cranes and Robots Using Genetic Algorithms as a Possible Mapping Approach"**, in Proc. of the IEEE International Conference on Robotics and Biomimetics (ROBIO), Shenzhen, China, 2013, pp. 643-650.

  This paper introduces a flexible and general control system architecture that allows for modelling, simulation and control of different models of maritime cranes and, more generally, robotic arms by using the same universal input device regardless of their differences in size, kinematic structure, degrees of freedom, body morphology, constraints and affordances. The manipulators that are to be controlled can be added to the system simply by defining the corresponding Denavit-Hartenberg table and their joint limits. The models can be simulated in a 3D visualisation environment, which provides the user with an intuitive visual feedback.

  The presented architecture represents the base for the research of a flexible mapping procedure between a universal input device and the manipulators to be controlled. As a case study, our first attempt of implementing such a mapping algorithm is also presented. This method is bio-inspired and it is based on the use of Genetic Algorithms (GA). Using this approach, the system is able to automatically learn the inverse kinematic properties of different models.

  Related simulations were carried out to validate the efficiency of proposed architecture and mapping method.

- **F. Sanfilippo**, L. I. Hatledal, A. Styve, H. Zhang and K. Y. Pettersen, **"Integrated Flexible Maritime Crane Architecture for the Offshore Simulation Centre AS (OSC)"**, accepted for publication to the IEEE Journal of Oceanic Engineering, 2015.

  The Offshore Simulator Centre AS (OSC) is the world's most advanced provider of simulators for demanding offshore operations. However, even though the OSC provides very powerful simulation tools, it is mainly designed for training purposes and it does not inherently offer any flexible methods concerning the control methodology. In fact, each crane model is controlled with a dedicated control algorithm that cannot be modified, accessed or replaced at run-time. As a result, it is not possible to dynamically switch between different control methods, nor is it possible to easily investigate alternative control approaches.

  To overcome these problems, a flexible and general control system architecture that allows for modelling flexible control algorithms of maritime cranes and more generally, robotic arms, was previously presented by our research group. However, in this previous work, a generic game engine was used to visualise the different models. In this work, the flexible and general control system architecture is integrated with the Crane Simulator developed by the OSC taking full advantage of the provided domain-consistent simulation tools. The Google *Protocol Buffers* protocol is adopted to realise the communication protocol. This integration establishes the base for the research of alternative control algorithms, which can be efficiently tested in a realistic maritime simulation environment.

  As a validating case study, an alternative control method based on particle swarm optimisation (PSO) is also presented. Related simulations are carried out to validate the efficiency of the proposed integration.

- **F. Sanfilippo**, L. I. Hatledal, H. Zhang, M. Fago and K. Y. Pettersen, **"JOpenShowVar: an Open-Source Cross-Platform Communication Interface to Kuka Robots"**, in Proc. of the IEEE International Conference on Information and Automation (ICIA), Hailar, China, 2014, pp. 1154-1159. **IEEE ICIA2014 Best Student Paper Award**.

  This paper introduces *JOpenShowVar*, a Java open-source cross-platform communication interface to *Kuka* robots that allows for reading and writing variables and data structures of the controlled manipulators. This interface, which is compatible with all *Kuka* robots that use *KR C4* and previous versions, runs as a client on a remote computer connected with the *Kuka* controller via *TCP/IP*. *JOpenShowVar* opens up to a variety of possible applications making it possible to use different input devices, sensors and to develop alternative control methods.

  To show the potential of the proposed interface, two case studies are presented. In

the first one, *JOpenShowVar* is used to control a *Kuka KR 6 R900 SIXX (KR AGILUS)* robot with an *Android* mobile device. In the second case study, the same manipulator is controlled with a *Leap Motion Controller* that supports hand and finger motions as input without requiring contact or touching. Related simulations are carried out to validate efficiency and flexibility of the proposed communication interface.

- **F. Sanfilippo**, L. I. Hatledal, H. Zhang, M. Fago and K. Y. Pettersen, **"JOpenShowVar: a Flexible Communication Toolbox for Controlling Kuka Robots"**, submitted to the IEEE Robotics & Automation Magazine, 2015.

  In this work, a more detailed overview of *JOpenShowVar* architecture is presented. Several new, more flexible and efficient, procedures are introduced in the latest release of the library to replace the old fundamental reading and writing method that is now deprecated. In addition to these new methods, some other high-level functions are also provided for reading angles and torques of the controlled manipulator. Some guidelines for allowing the user implementing new high-level procedures are discussed.

  Four case studies are presented to demonstrate the potential of *JOpenShowVar*. The first two case studies are open-loop applications, while the last two case studies describe the possibility of implementing closed-loop applications. In the first case study, the proposed interface is used to make it possible for an *Android* mobile device to control a *Kuka KR 6 R900 SIXX (KR AGILUS)* manipulator. In the second case study, the same *Kuka* robot is used to perform a two-dimensional line-following task that can be used for applications like advanced welding operations and similar. In the third case study, a closed-loop application is developed to control the same manipulator with a *Leap Motion Controller* that supports hand and finger motions as input without requiring contact or touching. In the fourth case study, a bidirectional closed-loop coupling is established between a *Force Dimension omega.7* haptic device and the same *Kuka* manipulator. Related experiments are carried out to validate the efficiency and flexibility of the proposed communication interface.

- **F. Sanfilippo**, L. I. Hatledal, H. Zhang, W. Rekdalsbakken and K. Y. Pettersen, **"A Wave Simulator and Active Heave Compensation Framework for Demanding Offshore Crane Operations"**, in Proc. of the IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), Halifax, Nova Scotia, Canada, 2015, pp. 1588-1593. **Third Prize at the Ocean Paper Poster Competition**.

  In this work, a framework is presented that makes it possible to reproduce the challenging operational scenario of controlling offshore cranes via a laboratory setup.

This framework can be used for testing different control methods and for training purposes. The system consists of an industrial robot, the *Kuka KR 6 R900 SIXX (KR AGILUS)* manipulator and a motion platform with three degrees of freedom. This work focuses on the system integration. The motion platform is used to simulate the wave effects, while the robotic arm is controlled by the user with a joystick. The wave contribution is monitored by means of an accelerometer mounted on the platform and it is used as a negative input to the manipulator's control algorithm so that active heave compensation methods can be achieved. Concerning the system architecture, the presented framework is built on open-source software and hardware. The control software is realised by applying strict multi-threading criteria to meet demanding real-time requirements.

Related simulations and experimental results are carried out to validate the efficiency of the proposed framework. In particular, it can be certified that this approach allows for an effective risk reduction from both an individual as well as an overall evaluation of the potential harm.

- **F. Sanfilippo**, L. I. Hatledal, Y. Chu, H. Zhang and K. Y. Pettersen, **"A Benchmarking Framework for Control Methods of Maritime Cranes Based on the Functional Mock-up Interface"**, submitted to the Springer Journal of Marine Science and Technology, 2015.

  In this work, a benchmark framework for advanced control methods of maritime cranes is presented. This framework is based on the use of the Functional Mock-up Interface (FMI), which is a tool independent standard for the exchange of dynamic models and for co-simulation. The system efficiently integrates different manipulator models, all the corresponding hydraulic systems, various vessels, and the surrounding environment for visualisation. Different control methods can be transparently implemented and tested.

  A set of routine tests, different cost functions, and metrics are provided – taking into account several factors, including position accuracy, energy consumption, quality, and safety for both the cranes and the surrounding environment. Each proposed routine test is task-oriented, and it systematically reproduces realistic on-board operation scenarios. The concept of operation profiles is introduced, allowing for defining different standard transporting and lifting operations. By considering task-oriented routines, this benchmark suite allows for comparing different control methods independently from the specific crane model to be controlled.

  Two alternative control methods for maritime cranes and robots are considered for an extensive comparison. The first method is based on the use of Genetic Algorithms (GAs), while the second method involves the use of Particle Swarm Optimisation (PSO). Related simulations are carried out to validate the benefits of the proposed benchmark suite.

CHAPTER 2

Design Methods and Control Architecture

In this chapter, our initial findings on developing efficient design methods and on the possibility of realising a flexible control architecture are presented.

With regard to modular robotic hands, the design of modular grasping grippers capable of adapting to different requirements and situations is investigated. A generalised modular model for robotic grasping is initially described. Some design guidelines are defined aiming to apply the principle of minimalism: choose the simplest configuration that will meet the requirements. Then, a novel algorithm that determines effective modular configurations to get efficient grasps of given objects is presented. This method allows for finding a trade-off between a simple gripper and more complex human like manipulators. Related simulations are carried out in order to test the proposed iterative approach. However, the proposed algorithm is only applied in a simulated environment and consequently it only allows for designing and testing virtual modular manipulators. To overcome this problem, *ModGrasp*, an open-source virtual and physical rapid-prototyping framework is successively proposed. *ModGrasp* allows for combining virtual and physical models by establishing a real-time one-to-one correspondence between them.

Regarding the study of maritime cranes and robotic arms, the low control flexibility and non-standardisation issues are addressed. In particular, a generalised manipulator model is initially considered. Then, based on this model, a flexible control architecture for different models of maritime cranes and robots is presented. Each manipulator can be controlled by using the same universal input device regardless of differences in size, kinematic structure, DOFs, body morphology, constraints and affordances.

Contributions of this chapter: considering modular robotic hands, the first contribution of this chapter consists of defining a generalised modular model for robotic grasping. This definition, together with several design guidelines, lays the foundation for developing an efficient design algorithm. To the best of our knowledge, the previous works on modular robot design are either bio-inspired [59] or generated by optimization procedure as, for instance, genetic algorithms [60]. However, very few of these works consider the principle of minimalism as fundamental design concept. Contrarily, we adopt this idea to propose an iterative design algorithm that allows for finding effective modular configurations to get efficient grasps of given objects. This is an important contribution because

this method makes it possible to design modular hands that use only the necessary number of DOFs to accomplish a given grasping task. Nevertheless, the proposed design method is initially developed as a simulation-based approach, thus it only allows for designing virtual modular manipulators. To overcome this limitation, we later propose *ModGrasp*, an open-source virtual and physical rapid-prototyping framework. This framework represents a significant contribution because it allows for establishing a real-time one-to-one correspondence between virtual and physical prototypes, opening up to a variety of possible application.

Respecting the study of maritime cranes and robotic arms, a generalised manipulator model is presented as the foundation for our designing and control studies in this specific field. Based on this model, a flexible control architecture for different manipulator models is introduced as one of the main contributions of this chapter. This contribution is particularly relevant because it represents the base for the research of flexible and transparent control methods.

Organization of this chapter: This chapter is systematically divided in two main sections. In Section 2.1, the challenge of designing and prototyping modular robotic hands is investigated. Similarly, in Section 2.2, a design approach is applied to the problems of low control flexibility and non-standardisation that currently affect the field of maritime cranes and robotic arms.

Publications: The results of this chapter concerning modular robotic hands are based on the papers [61], [62] and [63]. The findings related to maritime cranes and robotic arms are based on the paper P4.

2.1 Designing and Prototyping Modular Robotic Hands

In this section, a generalised modular model for robotic grasping is initially presented. The fundamental concept of modular robotic grasping is outlined and our guidelines for designing modular robotic robots are introduced. Based on these guidelines, an efficient simulation method for designing flexible grasping devices through an iterative procedure is depicted. Later on, *ModGrasp*, an open-source virtual and physical rapid-prototyping framework, is presented.

2.1.1 A Generalised Modular Model for Modular Robotic Hands

A generalised modular model is presented. The underlying idea consists in designing a modular device that can adapt its structure to the be object to grasp or to the task to be fulfilled. In other words, we define the guidelines for creating a device capable of adapting its structure and functionality to the characteristics of an object or a set of objects to be grasped. In doing this we want to respect the principle of minimalism: choose the simplest mechanical structure, the minimum number of actuators, the simplest set of sensors,

etc., that will do the job, or class of jobs. To formalise this idea, we introduce the concept of *modular grasping*.

Definition 2.1.1. The concept of *modular grasping* is used to indicate when identical modules are used to build linkages in order to realise the grasping functions. From a mechanical point of view, even if it is not the most efficient grasping approach, the modular grasping still meets the requirements of standardisation, modularisation, extendibility and low cost.

During our research work, two different modular robots have been used to build modular robotic hands. Initially, we adopted the *Y1* modular robot [64]. Successively a newly designed modular robot has been introduced. In the following, the same generalised modular model is successively considered based on the first and on the secondly adopted robots, respectively.

2.1.1.1 The Considered Generalised Model Built with the *Y1* Modular Robot

In our preliminary study, the *Y1* modular robot [64] with one DOF has been used as the basic element to build the modular device because of its versatility, robustness, low-cost and fast-prototyping features. A single *Y1* body module is 80 mm long, 50mm wide and 50 mm high. The dimension of the *Y1* module is not strictly comparable to the human phalanges. However, we decided to initially use this robot as building block in order to show the generality of our approach. Using docking blots, the modules can connect or disconnect easily and flexibly. Each joint is actuated by a RC servo. The *Y1* module is made of plastic so that the stiffness of its mechanical structure is quite low. We assume that each module has the same assembly selection to make the modular structure as simple as possible.

It should be noted that the *Y1* modular robot has mainly been used for building snake-like robots and testing their locomotion capabilities in previous literature [65]. However, based on this same robot, our research group previously proposed a snake-like configuration that introduces a task priority approach to manage both grasping and locomotion capabilities [66]. Nevertheless, Even if several robot configurations leading toward stable grasping have been outlined, the characteristics of snake-like robots are more suitable for Search and Rescue missions than for the manipulation of objects in human environments or industrial scenarios. Consequently, a generalised human-like modular model is considered in this work.

The generalised modular model consists of one or more chains of modules fixed on a base. Each module works as a chain link. A “proof of concept” is showed in Figure 2.1. The modules can be assembled to realize very dissimilar kinematic structures. Referring to a human-like hand, each chain can be considered as a finger, each module as a phalanx and the base as a palm.

The concept of modularity is also applied to the base of the proposed device model. In

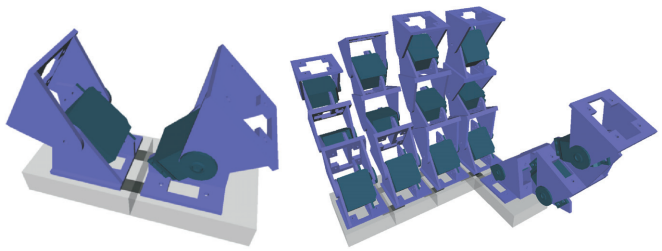


Figure 2.1: The modular grasping idea: thanks to its flexibility, the device can reproduce both simple grippers and more sophisticated kinematics like anthropomorphic robotic hands.

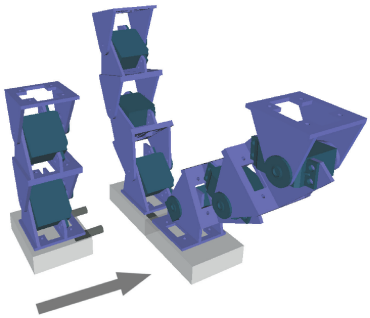


Figure 2.2: The concept of modular base: each finger has its own base plate that can be connected to form the gripper base.

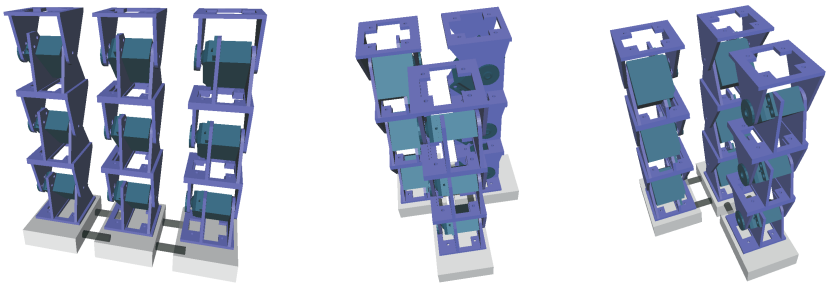
particular, each finger is attached on its own base plate module. The base plate modules of the fingers can be connected together using their predefined slots and hooks to form a unique base as shown in Figure 2.2.

Three possible modular base configurations have been defined as shown in Figure 2.3:

- *linear base*: finger opposition is avoided;
- *circular base*: the fingers are placed equally distant in a circle configuration;
- *opposable-fingers base*: one or more fingers are set to be opposable to the others.

This is a heuristic of the proposed approach since these three kinds of modular bases do not cover all possible gripper configurations. However, they are able to describe the most significant grasp models mimicking the human hand taxonomy, which is presented in [67].

The proposed model is very general and it can be extended to other types of modules with different characteristics and sizes. In this way, the modular structure can also allow



(a) (b) (c)

Figure 2.3: Possible base configurations for a three fingers modular device: *no finger opposition* (a), *circular* (b) and 1-*opposable-thumbs* (c).

the miniaturisation of the device. The reduction of the size, in fact, only depends on the building block characteristics, while the kinematic structure can be kept. This property of scalability can also be useful for dealing with objects of unknown size. In fact, the dimension of the device can change without affecting the proposed algorithm to determine the modular configuration.

2.1.1.2 The Considered Generalised Model Built with a Newly Designed Modular Robot

As already mentioned, the earlier adopted *Y1* modular robot was originally designed for building snake-like robots and testing their locomotion capabilities. As such, the *Y1* module does not provide properly designed contact surfaces for grasping objects. In addition, the size of the *Y1* module is considerably bigger when compared to the size of the human phalanges, thus not allowing to grasp human-sized objects.

To overcome these challenges, we introduced a newly designed modular robot, which is consistent with the guidelines of the previously considered generalised model, but it offers properly designed contact surfaces and it is smaller in size if compared to the *Y1* modular robot. In particular, the new fundamental building module is made by a standard micro servo motor and two metal brackets as shown in Figure 2.4-a. This elementary module, which works as a finger link, features one DOF and it is simple to construct and assemble, meeting the requirements of versatility, robustness, low-cost and rapid-prototyping. According to the rotation axis of the motor, each module can be connected to another one in a *pitch-pitch* or in a *pitch-yaw* connection configuration, as shown in Figure 2.4-b respectively. By assembling these modules, different finger configurations can be built. The concept of modularity is also applied to the base of the manipulator model. In particular, each finger is attached to a common base by means of two special brackets, which make abduction/adduction and flexion/extension movements possible, as shown in Figure 2.4-c. The base modules of the fingers can be connected together

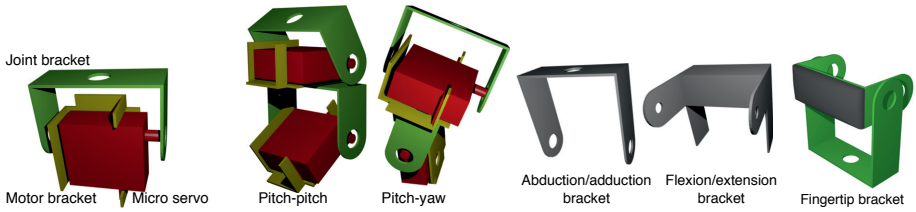


Figure 2.4: (a) the fundamental building module that is made by a standard micro servo motor and two metal brackets, (b) the *pitch-pitch* and *pitch-yaw* connection configurations respectively, (c) the two special brackets that allow for abduction/adduction and flexion/extension movements respectively, (d) the component that is used for the fingertips.

using their predefined slots and hooks to form a unique base as shown in the previous paragraph. Different base configuration can be achieved allowing for describing the most significant grasp models mimicking the human hand taxonomy. Moreover, to finalise the manipulator, a component, which is made by combining two joint brackets, is used for the fingertips, as shown in Figure 2.4-d. To increase friction, a small strip of soft material is taped to the fingertips with vulcanised tape. From a mechanical point of view, even though the design may not be the most efficient for grasping, our newly designed modular robot still meets the requirements of standardisation, modularisation, extendibility and low-cost.

2.1.2 An Efficient Design Method for Modular Grasping Hands

In this section, a new algorithm for modular grasping is presented for designing a flexible grasping device through an iterative procedure and considering human grasp quality values. It should be noted that the presented method is independent from the kind of modular robot adopted as fundamental building module.

In the following, the main variables of the algorithm are introduced. Let $m(i)$ be the total number of modules used for the modular gripper at the i-th iteration. Note that the base modules are not considered in this count. Let M be the maximum number of modules per finger of the modular device. M is computed at the beginning of the algorithm and depends on the features of the module and of the object to grasp. In particular, a lower bound of M is defined as follows:

$$M_{min} = \left\lceil \frac{R}{L} \right\rceil, \tag{2.1}$$

where R represents the radius of the minimum volume sphere that envelops the object to grasp and L is the length of one module. M_{min} takes into account the dimension of

the object to grasp. An upper bound of M is computed considering the maximum motor torque that can be exerted on the module. We considered as worst case the finger completely outstretched. In this situation, the maximum torque τ_{max} of the module closest to the finger base has to overcome the moment due to the weight w of the whole finger:

$$\tau_{max} > \frac{LMw}{2} \quad \Longrightarrow \quad M_{max} = \left\lfloor \frac{2\tau_{max}}{Lw} \right\rfloor . \tag{2.2}$$

Thereby, M is chosen as a trade-off between M_{min} and M_{max} during the initialization phase of the algorithm. Let $f_{min}(i)$ be the minimum number of fingers that must be considered in the device design to respect the limit M at the i-th iteration. The finger configuration can be denoted as follows:

$$\{x_1, x_2, ..., x_f\}, \tag{2.3}$$

where $x_j \in \mathbb{N}$ represents the number of modules of the j-th finger and f is the total number of fingers.

The goal of the proposed iterative procedure is to obtain a modular configuration that reaches a prefixed performance in terms of grasp quality using the least amount of modules possible. An evaluation of the grasp quality is thus required. The computation of grasp quality indices is known in the literature [68]. In this paper, the quality criteria introduced by Ferrari and Canny [58] is used. However, other solutions can be implemented without varying the algorithm structure. Ferrari and Canny considered a measure of the radius of the largest inscribed sphere centered at the origin that is contained in the so called *Grasp Wrench Space* (GWS) as quality index. The GWS is the set of all wrenches that can be resisted by a grasp if unit contact forces are applied at the contact points and it is given by the convex hull of the elementary wrenches:

$$GWS = ConvexHull\left(\cup_{i=0}^{n}\{w_{i,1}, \ldots, w_{i,k}\}\right), \tag{2.4}$$

where n is the number of contact points and k is the number of faces of the friction cone. The measure of the radius of the largest inscribed sphere centered at the origin that is contained in the GWS can be also seen as the magnitude of the largest worst-case disturbance wrench that can be resisted by a grasp with a unit strength grip. It will be hereafter denoted as Q, while the desired grasp quality will be denoted as $Q_{desired}$.

The flowchart of the proposed algorithm is shown in Figure 2.5. The main iterative loop starts with the simplest modular configuration which consists of one finger with one module and one base plate. With each iteration, an additional module is added to increase the possible DOFs. The number of modules for each finger is then set selecting one among all the possible gripper configurations which can be obtained considering $m(i)$ modules. Consequently, a configuration for the modular base of the device is selected, depending on the number of fingers, among the set of all the predefined base configurations. Once a configuration is generated, a grasp planner is used in order to find the best grasp achievable. If the corresponding grasp quality is less than $Q_{desired}$ and all the possible finger configurations and base configurations achievable with $m(i)$ modules have been tested, a new iteration begins and one more module is added.

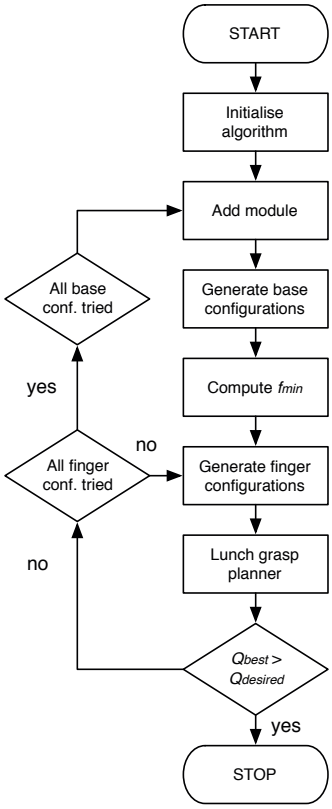


Figure 2.5: The flowchart of the proposed efficient design method for modular grasping hands.

In the following, the key steps of the algorithm are described.

Initialise algorithm. In this phase, the shape and the size of the *targetobject* is set. The values of M and $Q_{desired}$ are assigned. $m(0)$ and $f_{min}(0)$ are initialised, $m(0) = f_{min}(0) = 1$.

Generate base configurations. This step consists of defining the set of all the possible base configurations (*linear base*, *circular base*, *opposable-fingers base*). Note that other possible base configurations can be considered simply adding those in the predefined set.

Compute f_{min}. At each iteration a module is added, so $m(i) = m(i-1)+1$. The value of f_{min} has to be updated in order avoid the case of more than M modules per finger, so it can be defined as follows:

$$f_{min}(i) = \left\lceil \frac{m(i)}{M} \right\rceil . \tag{2.5}$$

Suppose that at iteration i, $m(i)$ is 3 and M is 3. The value of $f_{min}(i)$ is 1. At iteration

$i+1$, $m(i+1)$ is 4 so $f_{min}(i+1)$ is 2. This guarantees that it is not possible to have a configuration with only one finger with four modules respecting the limit M.

Generate fingers configurations. In this step a new configuration of the gripper is generated. The algorithm does not generate all the configurations at the same time. Each configuration is tested and a new one is generated only if $Q_{best} < Q_{desired}$. Otherwise the algorithm returns the current version. This approach avoids to test more configuration than those required.

Launch grasp planner. A grasp planner is used to determine the grasp quality achievable with each configuration for the given object. In general, the grasp planning problem can be solved in either the forward or the backward direction. In particular, in the proposed implementation of the algorithm, we used a forward solution implemented using the grasp planning simulator *OpenRAVE* [69]. A grasp is simulated by setting an initial base position (pose) and initial joint angles (pre-shape) to the manipulator device. For each gripper configuration fifty pose and pre-shapes are tested. Then, for each of them, the approach phase is realized by moving the device along the normal to the palm plane until it hits the target object. Hence, the fingers of the gripper close around the object until they can not close any more. The contacts between the device and the object are extracted, and the grasp quality index is calculated.

By the end of this step, the best grasp which can be obtained with the current configuration is returned together with the corresponding initial base position.

Note that any other planner, like for instance those implemented in *GraspIt!* [70], can be used without affecting the effectiveness of the proposed iterative algorithm.

Stop condition. The algorithm stops when the desired grasp quality is satisfied. The current modular configuration is efficient in the sense that it allows for reaching the desired grasp quality.

It should be noted that in this preliminary work, all the different configurations were initially manually built with *OpenRAVE*. Even though the effectiveness of the proposed design algorithm has been proven, the manual process of generating different modular configurations was a tedious and time-consuming task, which required a significant effort for the designer. To overcome this challenge, *OpenMRH* [71], a model generator for modular robotic hands was successively developed as a plugin for *OpenRAVE*. This plugin works as a *middleware* between the proposed iterative design algorithm and the *OpenRAVE* simulation environment.

2.1.2.1 Simulation Results

Related simulations are carried out in order to test the proposed iterative approach. In particular, the *Y1* modular robot is used as fundamental block for our simulations. A customised JAVA software plug-in is used to automatically generate all the possible modular

Table 2.1: Steps of the proposed design algorithm for the minimal configuration to grasp a ketchup bottle.

iteration	step	modular configuration	Q	time
$i=1$	1	$m=1$, $\{x_1=1\}$, -	0.0011	21s
$i=2$	2	$m=2$, $\{x_1=2\}$, -	0.0028	38s
	3	$m=2$, $\{x_1=1,x_2=1\}$, lin. base conf.	0.0459	49s
	4	$m=2$, $\{x_1=1,x_2=1\}$, circ. base or 1-opp.-finger conf.	0.0543	34s
	5	$m=2$, $\{x_1=1,x_2=1\}$, circ. base or 1-opp.-finger conf.	0.0613	35s
$i=3$	6	$m=3$, $\{x_1=3\}$, -	0.1270	47s

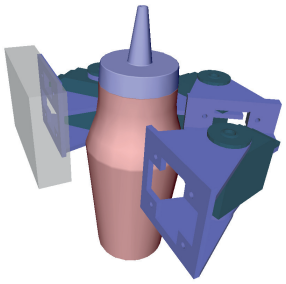


Figure 2.6: Minimal modular configuration to grasp a bottle of ketchup.

configurations according to the proposed algorithm. The grasping planner of *OpenRAVE* [69] is used to evaluate the grasp capability of each modular configuration.

The proposed design algorithm is used to find efficient modular configurations to grasp several daily objects. The maximum number of modules per finger M is set to 3. According to the experimental results presented in [72], the quality threshold $Q_{desired}$ is set to 0.1 since this or a greater measure of quality corresponds to grasps that a human would consider "stable".

For the sake of simplicity, only the example of a ketchup bottle is reported in detail. It can been observed that one finger with three modules is enough to reach the desired grasp quality. In Table 2.1 experiment details are reported. Note that for two fingers devices, *circular* and *1-opposable-finger* base configurations are the same. The reported value of Q refers to the best grasp obtained by each modular configuration at the i-th iteration. The listed execution times is obtained using an Intel i5 2.50GHz processor. The first modular configuration able to reach the desired grasp quality is shown in Figure 2.6.

Other simulations are performed in order to obtain effective configurations for grasping other objects or sets of objects. A phone, a book, a flask, a cup, a glass and an aircraft

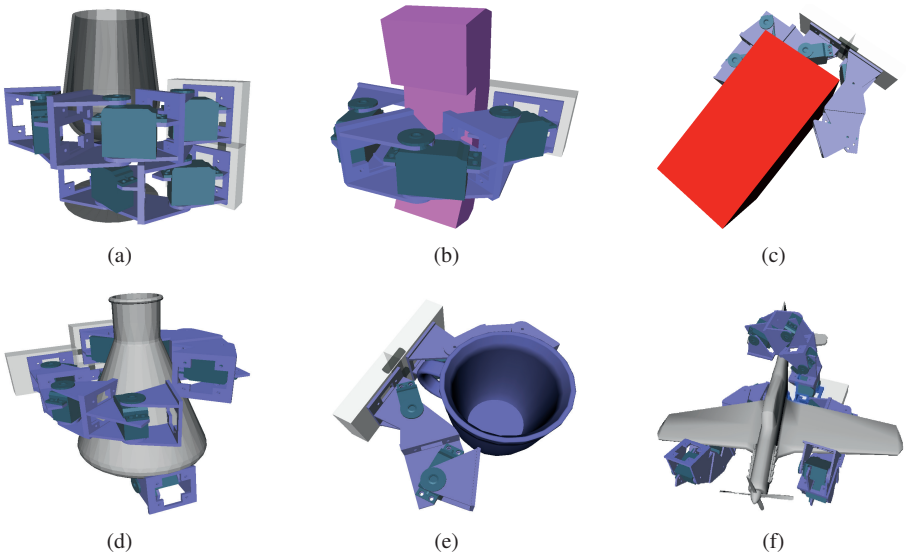


Figure 2.7: Efficient manipulator configurations for respectively grasping a glass (a), a phone (b), a book (c), a flask (d), a cup (e) and an aircraft model (f). For the latter object, the assembly selection of each module has been left as a free parameter in the design algorithm was tested.

Table 2.2: Efficient manipulator configurations for several daily objects.

target object	m	base configuration	fingers configuration	Q
glass	5	linear base conf.	$\{x_1 = 2, x_2 = 3\}$	0.12
phone	3	-	$\{x_1 = 3\}$	0.13
book	5	circ. or 1-opp.-finger base conf.	$\{x_1 = 2, x_2 = 3\}$	0.13
flask	9	circ. base conf.	$\{x_1 = 3, x_2 = 3, x_3 = 3\}$	0.14
cup	4	circ. or 1-opp.-finger base conf.	$\{x_1 = 2, x_2 = 2\}$	0.11
aircraft	14	circ. base conf.	$\{x_1 = 5, x_2 = 5, x_3 = 4\}$	0.53

model are tested. The resulting configurations are shown in Figure 2.7. Table 2.2 shows in detail the obtained modular configurations and the correspondent grasp qualities. For the aircraft model, the value of M is set to 5 and the possibility to leave the assembly selection of each module as a free parameter in the design algorithm is tested. The resulting modular configuration is quite different from classical grippers and is reported in Figure 2.7-f.

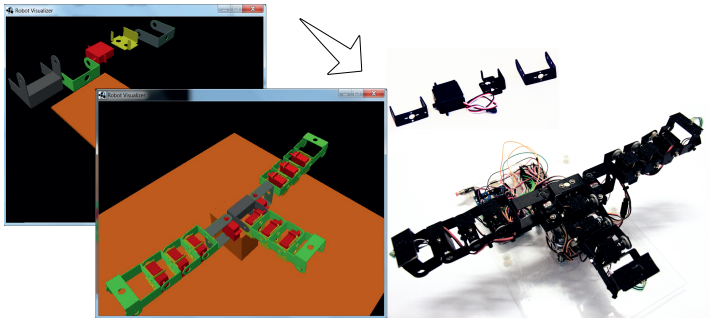


Figure 2.8: The idea of realising an integrated virtual and physical rapid-prototyping framework for the design, simulation and control of low-cost sensorised modular hands.

2.1.3 A Virtual and Physical Rapid-Prototyping Framework for Modular Robotic Hands

The design algorithm proposed in the previous section is a simulation-based approach and consequently it only allows for designing and testing virtual modular manipulators. To overcome this problem, a combined virtual and physical design framework is introduced in this section. When trying to combine the design for virtual and physical modular prototypes with different configurations, rapid-prototyping can be useful to significantly reduce the development time and to better analyse the main grasp properties. Based on this idea, *ModGrasp*, an open-source virtual and physical rapid-prototyping framework, is presented. The underlying idea is shown in Figure 2.8. The rapid-prototyping approach is combined with the modular concept so that different manipulator configurations can be rapidly modelled. This method consists of an immersive design process that involves mechanics, hardware and software. A real-time one-to-one correspondence between virtual and physical prototypes is established. The on-board, low-cost torque sensors provided within each module allow for evaluating the stability of the obtained grasps. An intuitive visual feedback is also provided during the designing phase by means of a 3-D visualisation environment. Moreover, both the virtual models and their physical counterparts can be controlled by using the same input device.

ModGrasp is an open-source project and it is available on-line at `https://github.com/aauc-mechlab/modgrasp`, along with several detailed class diagrams, all the mechanics, hardware schematics and demo videos.

The concept of modularity is applied to the framework architecture on both the software and hardware sides. In particular, as shown in Figure 2.9, a master-slave communication pattern is used. The system is based on a multi-threading and multi-level software paradigm. The control of each module is efficiently split in highly specialised processes (threads) that are hierarchically organised in different logical levels. Moreover, the controlled manipulators are simulated in a 3-D visualisation environment that communicates

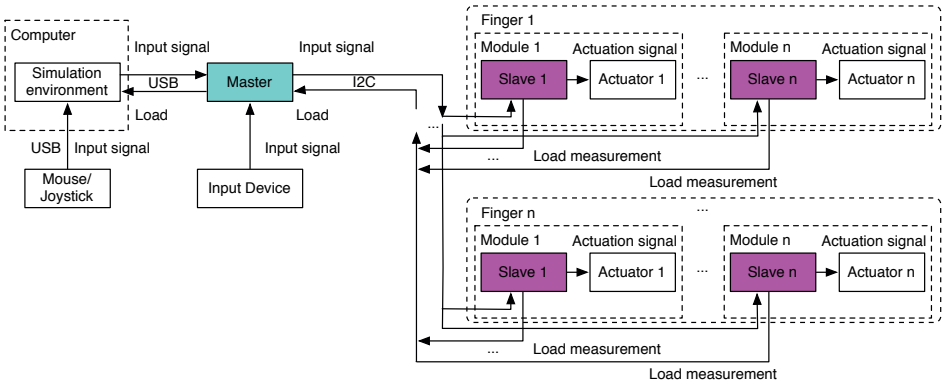


Figure 2.9: The *ModGrasp* master-slave communication pattern. Each module, or finger link, is independent, being controlled by one separate slave controller board, which directly communicates with the master controller board.

with the master controller. Each module is independent, being controlled by a self-reliant slave controller. In this way, the control architecture is:

- fully distributed, to support decentralised control and avoid single module failures (if one or more modules break or are disassembled from a prototype, the manipulator keeps working with the remaining functioning joints);
- dynamic, to be able to easily adapt to the topological changes and support different gripper configurations;
- scalable, to work for any configuration regardless of the shape and size. As such, a trade-off between simple grippers and more complex human-like manipulators can be reached.

In the following, the key elements of the newly designed framework are presented.

Simulation Environment. A basic simulation environment is used to simulate different manipulator prototypes on the computer side for free-hand motions. The virtual models to be controlled are added to the system simply by defining their corresponding standard Denavit-Hartenberg (D-H) tables [73]. In particular, this simulation environment is written in *Java*, thus making cross-platform support possible. To visualise the simulated behaviour, the *Open Graphics Library* (OpenGL) [74] is used on a low level by making raw *OpenGL-API* calls via the *Lightweight Java Game Library* (LWJGL) [75]. Moreover, to make dynamic creation and testing of different models possible without having to recompile the program each time, the simulator core is bound to the powerful, fast, lightweight and embeddable scripting engine *Lua* through the use of the *Kahlua* library [76].

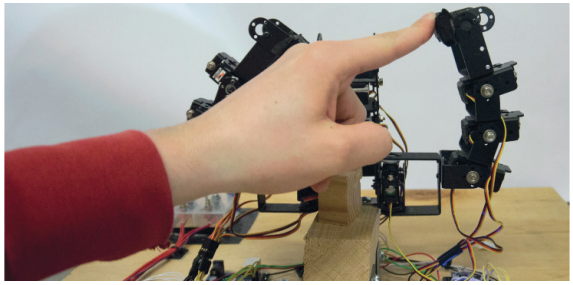


Figure 2.10: The sensitive collision detection approach: one of the fingers stops moving when an external force is applied.

Support for Different Input Devices. The modular manipulators can either be controlled directly from the simulator environment by means of a computer mouse/joystick or they can work stand-alone and be controlled by means of a different input device. In detail, in the first case, a real-time one-to-one correspondence between virtual and physical prototypes is established and the mouse/joystick DOFs are used as inputs. In the second case, an external input device is connected to the master board and is used to generate the input signal allowing the manipulator to be used without running the simulation environment. In both cases the input signal can be multi-dimensional. Thanks to the modularity of the framework architecture, any external input device can be used without influencing the effectiveness of the system. For instance, a set of potentiometer shafts can be used as input controller. Another possibility consists of using an EEG headset as input device. This topic is investigated in Chapter 3.

Controllers and Communication Protocol. On the hardware side, an *Arduino Uno* board [77] based on the *ATmega328* micro-controller is used as the master, while one *Arduino Nano* [77] board is used as a slave to control each module. *Arduino* is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. Using *Arduino* boards simplifies the amount of hardware and software development needed to get a system running. On the software side, *Arduino* provides a number of libraries to make programming the micro-controller easier. The choice of using *Arduino* boards makes the rapid-prototyping framework easy to maintain and makes it possible to add new features in the future.

The standard I^2C [78] is used as a communication protocol between the master and the slaves. In particular, each module has its own communication capacity. This protocol is chosen because it is relatively easy to set up and it also supports slaves having different addresses, thereby meeting the requirements of the framework architecture. In addition, the physical manipulator models communicate with the simulation environment through the serial interface of the master controller board.

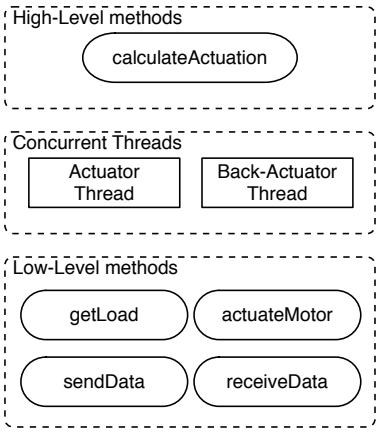


Figure 2.11: The multi-threading and multi-level hierarchical system of *ModGrasp*.

Low-cost Torque Sensing and Joint Compliance. In order to monitor the load of each joint actuator, the current is continuously measured from each slave controller. This feedback signal is very important in order to improve the manipulator dexterity. In particular, the current sensing at the joints level allows for a more accurate grasping of objects with different stiffness without squeesing or damaging them. By measuring the input current to each servo motor, the servo torque can be calculated and adjusted according to the task to be performed. Moreover, crucial functions like sensitive collision detection and compliant control actions are possible. In Figure 2.10, the sensitive collision detection is highlighted: one of the fingers stops moving when an external force is applied.

The basic motor equation is:

$$T = K_t I \sin(A), \tag{2.6}$$

where T is torque, K_t is the motor torque constant, I is the measured current and A is the angle between rotor and stator magnetic fields. In a properly-operating servomotor control scheme, A is held at 90 degrees and I is varied to meet the torque demands. It should be noted that this approach still meets the requirements while being very economical when compared to the use of traditional torque sensors.

Multi-threading and multi-level hierarchical system. Each module is controlled by a corresponding slave controller that runs a multi-threading and multi-level control program, as shown in Figure 2.11. Three different levels are defined for the control pattern of each module:

- the *Low-Level methods* layer includes the low-level functions that are used to actuate the motor (*actuateMotor*), to sense the motor load (*getLoad*), and to communicate with the master (*sendData* and *receiveData*);

- the *Concurrent Threads* level is the layer where the concurrent processes are implemented. This level can access both the *Low-Level methods* as well as the *High-Level methods*;
- the *High-Level methods* layer includes the high-level and distributed control function, *calculateActuation*, which determines the joint actuation according to the adopted control method.

To improve the performance of the proposed architecture, a multi-threading pattern is adopted. Essentially, two concurrent processes are considered:

- the *Actuator Thread* takes care of the motor actuation by calling the underlying *actuateMotor* method;
- the *Back-Actuator Thread* is responsible for the joint back-actuation when the servo load reaches a predefined threshold. The motor is programmed to step back slightly (according to a predefined step-back value) in order to reduce the torque.

From an implementation point of view, the *mthread* library [79], which is an Arduino-compatible multi-threading library, is chosen.

Control Approach. The possibility of implementing certain control features does not influence the design for the proposed prototyping framework. In particular, because of the modularity properties of the framework architecture, the user can implement different control algorithms for the models according to current needs.

Conventional robotic control design tools and equations may be sufficient for simple prototypes with a low number of DOFs, but when the complexity of the modular model increases, a highly flexible and general control algorithm is needed. This can happen in the case of an increase in the number of DOFs or when different modular configurations must be controlled independently of their specific morphology. This challenge is described in Chapter 3. Related simulation results are also shown in Chapter 3.

2.2 Designing a Flexible Framework for Maritime Cranes and Robotic Arms

In the previous sections, the field of modular robotic hands has been investigated from a design point of view. Similarly, in this section, a generalised manipulator model is initially considered concerning maritime cranes and robotic arms. Then, based on this model, a design approach is applied to the problems of low control flexibility and non-standardisation that currently affect the field of maritime cranes and robotic arms. In particular, a flexible control architecture for maritime cranes and robotic arms is presented.

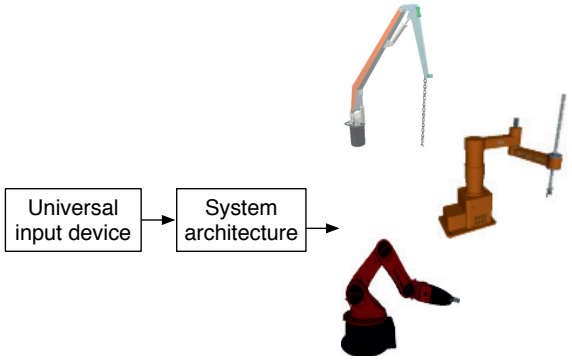


Figure 2.12: The idea of realising a control architecture that allows for simulating and operating different models of maritime cranes and, more generally, robotic arms by using the same universal input device.

2.2.1 A Generalised Manipulator Model for Maritime Cranes and Robotic Arms

A generalised manipulator model is considered. The generalised model consists of a kinematic chain that can be controlled by setting the position or the velocity of the joints.

From a kinematic point of view, the end-effector of an offshore crane usually consists of a wire which is used to lift and transfer objects, while robotic arms are commonly equipped with more complex devices like grippers or tools. In a wider sense, in both cases, the end-effector can be seen as the part of the manipulator that interacts with the work environment and it may be modelled as part of the same kinematic chain. The proposed architecture, however, also allows the end-effector to be modelled as a distinct sub-chain that can be controlled separately. So, in general, a mapping control method may or may not consider the control of the whole manipulator. Decoupling the model of the end-effector from the model of the manipulator can, in some cases, greatly simplify the mapping control algorithm since the complexity of the system generally increases more than linearly with the number of DOF.

2.2.2 A Flexible Control Architecture for Maritime Cranes and Robotic Arms

Based on the previously introduced generalised manipulator model, a general architecture is presented that allows for modelling, simulation and control of different models of maritime cranes and, more generally, robotic arms by using the same universal input device. The idea is shown in Figure 2.12. The main challenge of doing this consists of finding a flexible mapping procedure to map the fixed degrees of freedom of the universal input

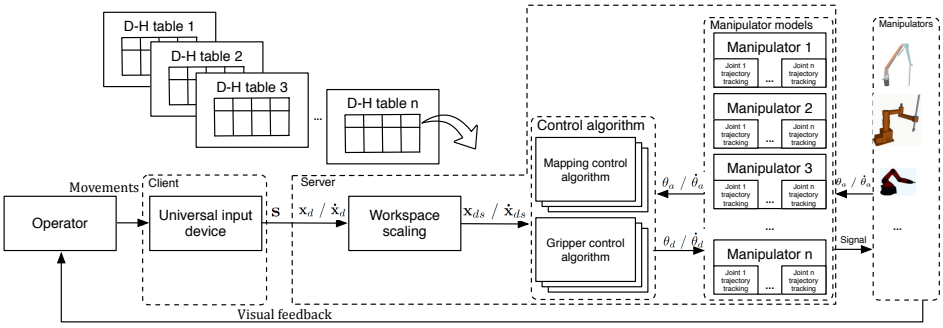


Figure 2.13: The proposed control system architecture for operating different maritime cranes and robotic arms.

device to the variable degrees of freedom of the cranes or robots to be controlled. This process has to be realised regardless of their differences in size, kinematic structure, body morphology, constraints, affordances and similar. The presented architecture allows for designing and testing different mapping procedures.

System Architecture. The proposed control system architecture is shown in Figure 2.13. It is a client-server architecture with the input device running as a client and communicating with a server where the logic of the control algorithm is implemented. The controlled arms are simulated in a 3-D visualisation environment, which also acts as a client and provides the user with an intuitive visual feedback.

The proposed architecture provides the possibility of controlling the arms in position mode or velocity mode. The user experience is substantially different in each case. When using the position control mode, the operator simply controls the position of the tip of the crane with constant velocity; when operating in velocity control mode, the operator also sets the velocity of the end-effector by using the universal input device. In the first case, when the operator releases the input device, the tip of the crane moves back to its starting point, while in the second scenario, the crane just stops moving but it keeps the last given position. To realise these two possible operation modes, when the operator manoeuvres the manipulator, a vector signal with no semantic, $\mathbf{s}$, is sent from the universal input device to the server. Here, according to the operational scenario, the vector signal is interpreted as the desired position $\mathbf{x}_d$ or the desired velocity vector $\dot{\mathbf{x}}_d$.

Additionally, in order to adjust the size of the input device's workspace to the arm to be controlled, a scaling factor is introduced to calculate the coordinate of the point to be reached. The proposed architecture allows for expanding and shifting the small-scale physical workspace of the input device to a virtual expanded workspace allowing the robot arm for more accurate and precise movements. In particular, referring to Figure 2.14 and denoting the reference frame of the input device's physical workspace with O_i, the

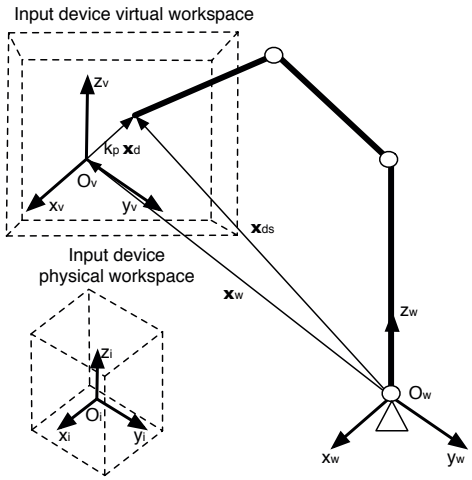


Figure 2.14: Input device physical and virtual workspaces for a manipulator to be controlled.

reference frame of the input device's virtual workspace with O_v, and the reference frame of the manipulator workspace with Ow, the desired scaled position, $\mathbf{x}_{ds}$, is calculated as follows:

$$\mathbf{x}_{ds} = k_p \mathbf{x}_d + \mathbf{x}_w, \tag{2.7}$$

where k_p is the position scaling factor and $\mathbf{x}_w$ is a shifting vector that defines the position of the virtual reference frame with respect to the global reference frame. Similarly, the desired velocity vector can also be scaled to allow the operator to execute slower or faster movements according to the task to be accomplished. The desired scaled velocity vector, $\dot{\mathbf{x}}_{ds}$, can be obtained as follows:

$$\dot{\mathbf{x}}_{ds} = k_v \dot{\mathbf{x}}_d, \tag{2.8}$$

where, k_v is the velocity scaling factor.

Then, according to the desired mode of operation, the mapping control algorithm parses those values to the desired joint angles θ_d or desired joint velocities $\dot{\theta}_d$ of the manipulator, respectively. Essentially, for all the different models to be controlled, the mapping methods have to implement the classic IK functions that can be generalised as follows:

$$\theta_d = f_p^{-1}(\mathbf{x}_{ds}), \tag{2.9}$$

concerning position control, and

$$\dot{\theta}_d = f_v^{-1}(\theta_a, \dot{\mathbf{x}}_{ds}), \tag{2.10}$$

for velocity control, where θ_a is the the actual joint angles vector.

The calculated desired joint angles θ_d or joint velocities $\dot{\theta}_d$ are then forwarded from the server to the visualisation environment in order to actuate the crane model. As feedback

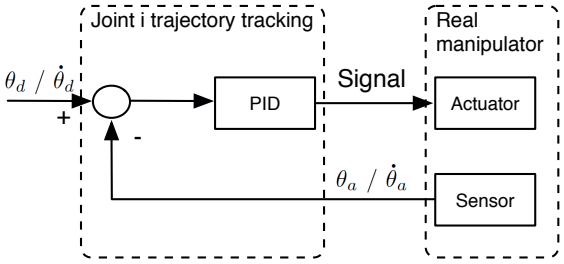


Figure 2.15: Trajectory tracking using a PID controllerm for a manipulator to be controlled.

from the visualisation environment, the actual joint angles θ_a and joint velocities $\dot{\theta}_a$ are sent back to the server and can be used by the control mapping algorithm.

Notice that the proposed architecture allow for implementing different mapping methods. Each mapping control algorithm has to realise the mapping between the fixed DOFs of the universal input device and the variable DOFs of the manipulator to be controlled. It is important that each control algorithm be implemented as an independent and interchangeable module and that it satisfies the interface specified by the system, (2.9) and (2.10), in order to respect the modularity of the proposed architecture.

A relevant feature of the proposed architecture is that the robot model can be separated from the control algorithm to be used. In particular, no matter which control algorithm is used, the manipulators to be controlled can be added to the system simply by defining their corresponding standard Denavit-Hartenberg (D-H) tables [73] and their joint limits.

For all the models to be controlled, the different mapping methods calculate the corresponding sampling point configurations for the desired end-effector's positions. In other words, each mapping method works as a motion planner. In order to ensure smooth movements for the manipulators it is necessary to generate trajectories out of these given sampling points. A well-suited trajectory is the basic prerequisite for the design of a high-performance tracking controller and ensures that no kinematic nor dynamic limits are exceeded. Such a controller guarantees that the controlled robot will follow its specified path without drifting away. Therefore, feedback control has to be applied to be able to compensate external disturbances as well as disturbances from communication time delays. Note that time data is a free parameter because the sampling time of the mapping algorithm is generally not constant.

A possible solution for generating well-suited trajectories consists of using a Proportional Integral Derivative (PID) controller for each joint, as shown in Figure 2.15. Notice that using this approach, the nature of the crane actuators - whether they are hydraulic, electric or mechanical - can be also taken into account.

Implementation Details. From an implementation point of view, the logic of the control architecture lies on the server side, which is implemented by using the *Java* programming language. Each manipulator to be controlled is modelled as a *Java* class which embodies a D-H table, a set of joints, a workspace as attributes and a *Solver* as an abstract subclass. The *Solver* abstract subclass has two methods - *positionSolver* and *velocitySolver* - which have the prototypes that the mapping functions have - (2.9) and (2.10) respectively. The GA mapping method described in the previous section is a particular implementation of this *Solver* but new mapping methods can easily be added by simply providing a corresponding implementation of the same abstract subclass.

To speed up the developing process and to improve the reliability of the system, several libraries were used. In particular, the *Efficient Java Matrix Library* [80] was adopted to add support for matrix manipulations. Moreover, the manipulators to be controlled can easily be added to the system by simply defining their corresponding D-H tables and their specific joint limits in a *XML* document.

Regarding the visualisation environment, in this preliminary work, the game engine *Unity3D* [81] was used to visualise the different models. However, any other visualisation environment could be used without affecting the effectiveness of the proposed architecture.

Based on the proposed flexible architecture, different flexible mapping methods are presented in Chapter 3. Related simulation results are also presented in Chapter 3.

Chapter Summary

In this chapter, we presented our preliminary studies on developing efficient design methods and on the possibility of realising a flexible control architecture for robotic manipulators.

Our initial finding in the field of modular robotic hands are first summarised in the following.

- Different design guidelines were proposed for creating a modular device capable of adapting its structure and functionality to the characteristics of an object or a set of objects to be grasped. In doing this, the principle of minimalism was adopted: choose the simplest mechanical structure, the minimum number of actuators, the simplest set of sensors.
- To formalise this idea, the concept of *modular grasping* was introduced to indicate when identical modules are used to build linkages in order to realise the grasping functions. From a mechanical point of view, even if it is not the most efficient grasping approach, the modular grasping still meets the requirements of standardisation, modularisation, extendibility and low cost.
- A generalised modular model for modular robotic hands was presented. This general model was built by using two different modular robots. Initially, we adopted the

Y1 modular robot. Successively, a newly designed modular robot was introduced.

- Based on the proposed generalised model, an effective design algorithm was presented for designing flexible grasping devices through an iterative procedure and considering human grasp quality values. In particular, the novel algorithm allows for determining effective modular configurations to get efficient grasps of given objects. The resulting modular configurations are able to perform effective grasps that a human would consider "stable". Related simulations were carried out with several daily objects proving the effectiveness of the proposed iterative approach.
- However, the proposed design algorithm is a simulation based approach, therefore it only allows for designing and testing virtual modular manipulators. To overcome this challenge, *ModGrasp*, a combined virtual and physical design framework was introduced. The system is built on open-source software and it combines the rapid-prototyping approach with the modular concept so that different manipulator configurations can be rapidly modelled. The framework makes it possible to realise an immersive design process that involves mechanics, hardware and software. A real-time one-to-one correspondence between virtual and physical prototypes is established. The on-board, low-cost torque sensors provided within each module allow for evaluating the stability of the obtained grasps. An intuitive visual feedback is also provided during the designing phase by means of a 3-D visualisation environment. Moreover, both the virtual models and their physical counterparts can be controlled by using the same input device.

Similarly, our preliminary results pertaining to maritime cranes and robotic arms focused on addressing design challenges. Our initial findings are listed in the following.

- Emphasising the similarities between maritime cranes and robots, a generalised manipulator model was initially considered. The generalised model consists of a kinematic chain that can be controlled by setting the position or the velocity of the joints.
- Based on the proposed generalised manipulator model, a general architecture was presented that allows for modelling, simulation and control of different models of maritime cranes and, more generally, robotic arms by using the same universal input device. The proposed architecture is a client-server architecture with the input device running as a client and communicating with a server where the logic of the control algorithm is implemented. The controlled arms are simulated in a 3-D visualisation environment, which also acts as a client and provides the user with an intuitive visual feedback. The presented architecture represents the base for the research of a flexible control procedure between a universal input device and the manipulators to be controlled.

CHAPTER 3

Alternative Control Algorithms

In this chapter, we present different alternative algorithms that are able to scale and control different manipulators regardless of differences in size, kinematic structure, DOFs, body morphology, constraints and affordances.

With regard to modular robotic hands, a novel control method is presented. The adoption of this control method is motivated by the fact that, despite the simplicity of a modular manipulator model, with the increase in the number of its fingers and modules, it becomes rival to the human hand in terms of complexity. To deal with this challenge, a biologically-inspired control method is adopted. This method is inspired to the human hand not only with regard to size and configuration, but also as regards to the control. The proposed method is implemented based on *ModGrasp*, our virtual and physical rapid-prototyping framework for modular robotic hands which was previously presented in Chapter 2. As a case study, a mind-controlled, low-cost modular manipulator is presented.

Similarly, the possibility of developing alternative control methods is also investigated concerning the field of maritime cranes and robotic arms. In this regard, two alternative control methods are presented aiming to transparently control different manipulators and configurations. The first method is based on the use of Genetic Algorithms (GAs) [33]. The second method involves the use of Particle Swarm Optimisation (PSO) [39]. Both the proposed control methods are implemented based on the flexible control architecture for maritime cranes and robot which was previously presented in Chapter 2.

Contributions of this chapter: with regard to modular robotic hands, the first contribution of this chapter is a novel control method, which is based on a biologically-inspired idea. Notably, this method adopts human synergies to control the models, as described in [56]. The adoption of this method is relevant for this thesis because it allows to transparently control different modular hand configurations independently from their number of DOFs. Several relevant application scenarios are possible.

Concerning the fields of maritime cranes and robotic arms, our contribution consist of developing two alternative control methods that aim to transparently control different manipulators and configurations. These two methods are based on the use of GAs and PSO, respectively. Both the proposed control methods are relevant for this thesis because they do not assume a priori knowledge for the IK models. In particular, these algorithms derive

the kinematic properties of different robotic manipulators by adopting novel optimisation methods.

Organization of this chapter: Following the same line of the thesis, this chapter is divided in two main sections. In Section 3.1, a biologically-inspired control for modular robotic hands is described. Based on this alternative control method, a mind-controlled, low-cost modular manipulator is presented as a case study. Similarly, in Section 3.2, we present two alternative algorithms that allows for controlling different maritime cranes and, more generally, robotic arms. These methods are based on the use of machine learning procedures. In particular, the first method uses GAs, while the second algorithm is based on the use of PSO.

Publications: The results of this chapter concerning modular robotic hands are based on the papers [62] and [63]. The findings related to maritime cranes and robotic arms are based on the papers [82], and [83].

3.1 Alternative Control Algorithms for Controlling Modular Robotic Hands

When considering a possible control method for simple modular robotic hands with a low number of DOFs, it may be sufficient to use conventional robotic control design tools and equations [84]. However, when the complexity of the modular grasping model increases in terms of DOFs or when different modular configurations must be controlled independently of their specific morphology, a highly flexible and general control algorithm is needed. In this section, the challenge of developing such a flexible control algorithm for modular robotic hands is addressed. In the following, a biologically-inspired control method is presented.

3.1.1 A Synergistic Control Method for Modular Robotic Hands

There is a need for a transparent alternative to control modular robotic hands independently from their specific structure, and focusing only on the task. Such a general control approach needs to be flexible to effectively deal with a variable modular configuration and number of DOF. A deeper understanding of how the brain exploits the high redundancy of human hands could be an important key in the development of such a control algorithm. In particular, some studies demonstrate that, despite the complexity of the human hand, a few variables are able to account for most of the variance in the patterns of all the possible configurations and movements [85]. These same studies showed that the first two principal components account for most of the variability in the data, more than 80% of the variance in the hand postures. In this context, the principal components were referred to as *synergies* given that, in the *sensorimotor* system of the human hand, combined actions are favoured over individual component actions, with advantages in terms

of simplification and efficiency of the overall system. Intuitively, this reduction of DOFs can be used to decrease the complexity of the control algorithm for robotic hands with an anthropomorphic structure that closely copies the structure of the human hand. Nonetheless, several approaches for mapping the human hand *synergies* to differently structured robotic hands have been presented [86], [87], showing that this idea is feasible.

The key equations necessary to study manipulators controlled by *synergies* from a kinematic point of view are briefly introduced here. A more detailed presentation of the problem is described in [56]. According to a model inspired by human hand *synergies*, we suppose that the hand is actuated using a number of inputs whose dimension is lower than the number of hand joints. In particular, let the manipulator be described by the joint variable vector $\mathbf{q}_h \in \Re^{n_{q_h}}$, with n_{q_h} representing the number of actuated joints. We assume that the subspace of all configurations can be represented by an input vector of a lower dimension $\mathbf{z} \in \Re^{n_z}$ (with n_z denoting the number of inputs and $n_z \leq n_{q_h}$) which parameterises the motion of the joint variables along the *synergies*. In terms of velocities, one gets:

$$\dot{\mathbf{q}}_h = \mathbf{S}_h \dot{\mathbf{z}}, \tag{3.1}$$

being $\mathbf{S}_h \in \Re^{n_{q_h} \times n_z}$ the synergy matrix.

To show the potential of this alternative control method, a case study is presented in the following.

3.1.1.1 A Mind-Controlled, Low-Cost Modular Manipulator

A mind-controlled, three-fingered modular manipulator is presented as a case study. In particular, *ModGrasp*, the open-source virtual and physical rapid-prototyping framework that was presented in the previous chapter, is used to implement the proposed alternative control method. The underlying idea is shown in Figure 3.1. The user's levels of attention and meditation are used to generate a two-dimensional inputs to control the hand. Since the manipulator features 11 DOFs, a synergistic control approach is chosen to map inputs with outputs with such a different dimensionality. A demo video of this case study is available on-line at `http://youtu.be/2CIYboez9r0`.

EEG Control Input. The previously introduced framework of *ModGrasp* allows for using different input devices to control the modular device. In this case study, an EEG headset is used as input device. In particular, the *NeuroSky MindWave* headset [88] is adopted. This EEG headset is wirelessly connected to the master board by using the Bluetooth protocol and is used to generate the input signal allowing the manipulator to be used with or without running the simulation environment. The choice of using an EEG headset is motivated by the wish of the authors for providing a more intuitive control interface to the developed prototypes. This choice opens up to a variety of possible applications including the development of modular prototypes for mind-controlled prosthetic hands.

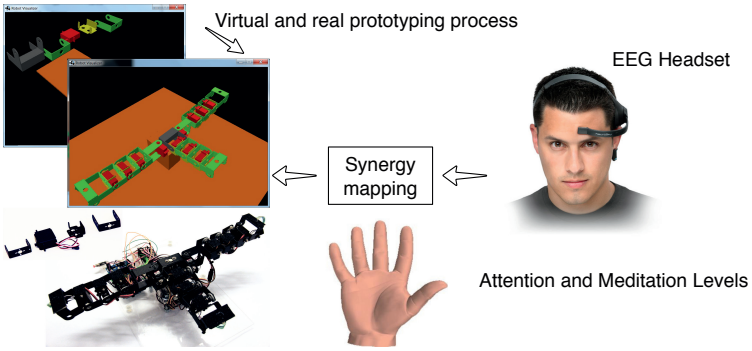


Figure 3.1: A mind-controlled, three-fingered modular manipulator is presented.

The human brain is made up of billions of interconnected neurons. As neurons interact, patterns manifest as singular thoughts such as a math calculation, and broad emotional states such as attention. Every interaction between neurons creates a miniscule electrical discharge, measurable by EEG machines. By themselves, these charges are impossible to measure from outside the skull. However, a dominant mental state, driven by collective neuron activity created by hundreds of thousands concurrent discharges, can be measured. Different brain states are the result of different patterns of neural interaction. These patterns lead to waves characterised by different amplitudes and frequencies. As examples, brainwaves between 12 and 30 hertz, *Beta Waves*, are associated with concentration, while waves between 8 and 12 hertz, *Alpha Waves*, are associated with calm relaxation.

In the proposed architecture, the user's *Alpha Waves* and *Beta Waves* are monitored by using an EEG headset and used to generate the input signal. The control objective is that when the user start focusing, for instance by starting reading or doing some simple math calculation, the levels of attention and meditation increase causing the controlled manipulator to close the fingers so that grasping operations can be achieved. Once the user loses focus, the controlled hand will open up the fingers again and release the grasped object. The control objective idea is shown in Figure 3.2.

Manipulator Model. A three-fingered modular hand is adopted in this case study to be controlled with an EEG headset. In Figure 3.3, the hand model is shown. It consists of a 3 DOFs thumb, which opposes the other two fingers, each having 4 DOFs. Table 3.1 is the D-H table of the thumb, whereas Table 3.2 is the D-H table of the other two fingers. The fingers are directly attached to a wooden plate that is used as a base. This particular configuration is chosen for simplicity and as a heuristic design in order to accurately describe the most significant grasping models that mimic human hand taxonomy, as outlined in [89]. The *synergy* matrix, determined by following the mapping approach proposed in [87], is:

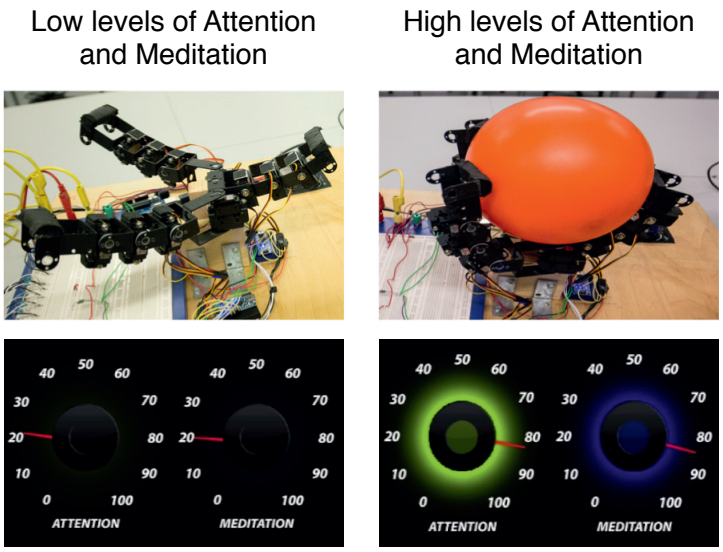


Figure 3.2: The control objective idea: by monitoring the user's *Alpha Waves* and *Beta Waves*, successful grasping operations can be performed.

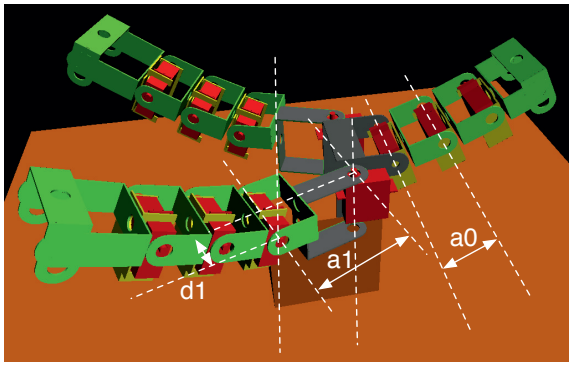


Figure 3.3: The three-fingered modular manipulator and the corresponding virtual model. The parameters $a_0 = 3.2cm$, $a_1 = 5.9cm$ and $d_1 = 1.5cm$ are used to determine the D-H tables reported in Table 3.1 and Table 3.2.

Table 3.1: D-H table of the thumb, where $a_0 = 3.2cm$ is shown in Figure 3.3.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	a_0	0	θ_1
2	0	a_0	0	θ_2
3	0	a_0	0	θ_3

Table 3.2: D-H table of the other two fingers, where $d_1 = 1.5cm$ and $a_1 = 5.9cm$ are shown in Figure 3.3.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	d_1	θ_1
2	$\frac{\pi}{2}$	a_1	0	θ_2
3	0	a_0	0	θ_3
4	0	a_0	0	θ_4

$$\mathbf{S}_h = \begin{bmatrix} -0.7 & 0 \\ -0.2 & 0 \\ -0.1 & 0 \\ 0 & -1.6 \\ -0.7 & 0 \\ -0.2 & 0 \\ -0.1 & 0 \\ 0 & 1.6 \\ -0.7 & 0 \\ -0.2 & 0 \\ -0.1 & 0 \end{bmatrix} \begin{matrix} \} Thumb \\ \\ \\ \\ \} Finger1 \\ \\ \\ \\ \} Finger2 \\ \\ \end{matrix} \tag{3.2}$$

In this particular case, an input vector, $\mathbf{z} \in \Re^2$, is used to select the first two principal *synergy* components. This input vector contains the two mapped signals coming from the EEG headset monitoring the levels of attention and meditation. Related experiments and results are presented in the following.

3.1.1.2 Experiment Results

Related simulations are carried out in order to test the presented synergistic control method within the particular case study of the three-fingered modular manipulator. Particularly, a balloon is selected for use in performing a grasp and release experiment, as shown in Figure 3.4. This experiment aims to perform a grasp that a human would consider "stable". To achieve such a kind of task, extensive training is required for the user in order to efficiently control the appropriate attention and meditation levels. The time plots of the EEG inputs and of the corresponding estimated torque values for the joints while grasping and

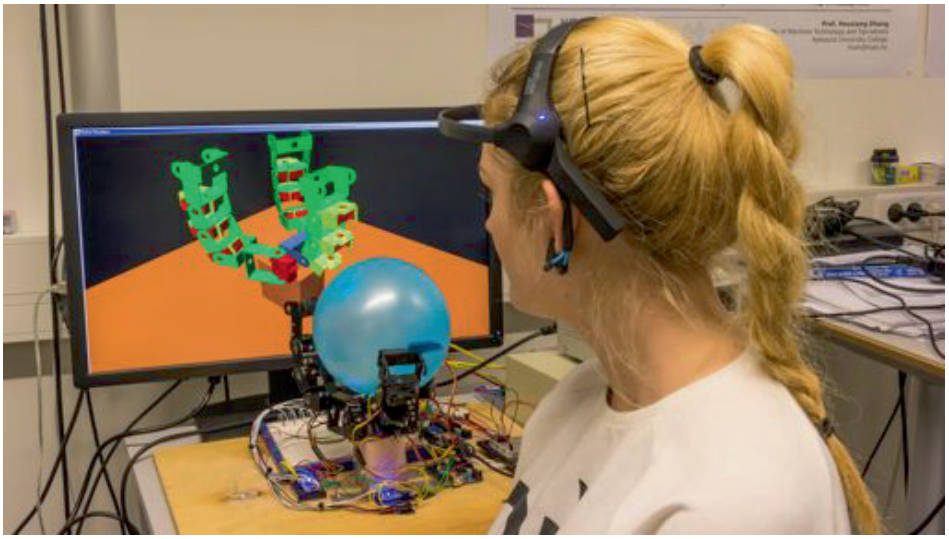


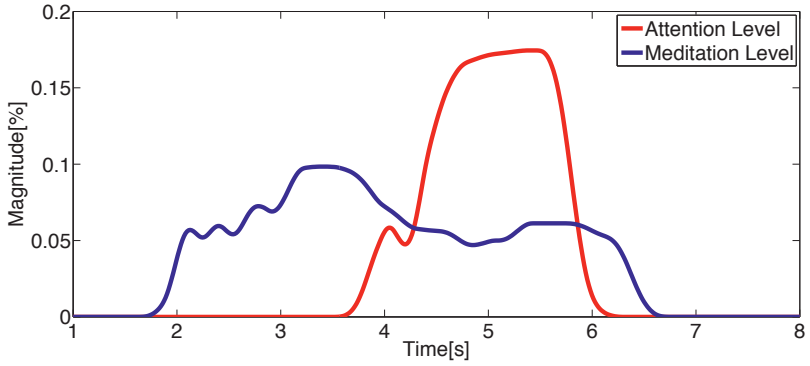
Figure 3.4: A balloon is selected for use in performing a grasp and release experiment with a mind-controlled, low-cost modular manipulator.

releasing the balloon are shown in Figure 3.5-a and in Figure 3.5-b, respectively. Note that a low-pass filter is applied to reduce the noise from the collected data. As expected, the torque values increase when the contact is made with the object to be grasped. In addition, symmetric patterns in the torque values can be seen between symmetric joints while executing the grasp. It should be noted that grasping a balloon is a particularly challenging task for a robotic hand. With this experiment, our framework demonstrates effectiveness in designing hands that are capable of performing such a kind of task with sufficient dexterity.

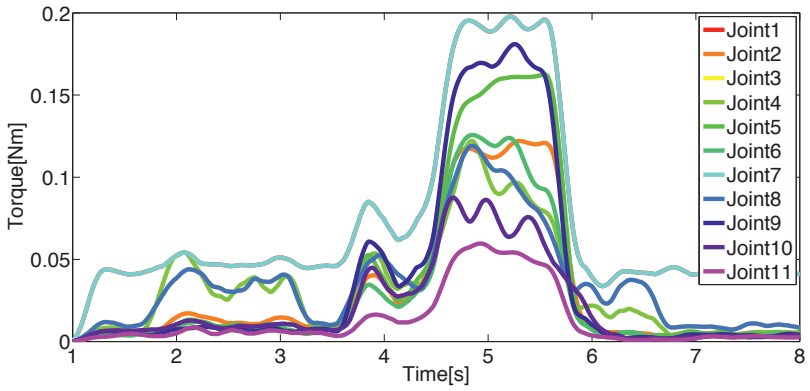
3.2 Alternative Control Algorithms for Controlling Maritime Cranes and Robotic Arms

In the previous sections, the possibility of developing alternative control algorithms in the field of modular robotic hands has been investigated. Similarly, in this section, the possibility of implementing alternative control methods for controlling maritime cranes and, more generally, robotic arms is considered.

When designing a control algorithm for a crane or for a robotic arm, it is necessary to determine the kinematic properties of the system. One possible approach consists in studying the IK model of the manipulator to be controlled. This common approach allows for



(a)



(b)

Figure 3.5: (a) the time plot for the EEG inputs showing the attention and the meditation levels, (b) the corresponding estimated torque values for the joints while grasping and releasing the balloon.

either introducing analytical methods, which offer exact solutions for simple kinematic chains, or for proposing solutions based on numerical methods. Nonetheless, the inverse kinematics may have multiple solutions when considering arms with redundant DOFs and singularity problems could arise. Moreover, this method could hardily be applied when considering to control different manipulators using a universal input device because several IK models would be needed: one for each arm or crane to be controlled. An alternative solution to the problem might consist of using methods that derives the kinematic properties by applying an optimisation method or a machine learning procedure. In this way, the system would be able to automatically obtain the kinematic properties of different arms, and new models could also easily be added.

In the following, two alternative control methods are considered, one based on the use of GAs and the other one based on the use of PSO. Both the proposed control methods are implemented based on the flexible control architecture for maritime cranes and robot which was previously presented in Chapter 2.

3.2.1 An Alternative Control Algorithm Based on GAs

A genetic algorithm is a search heuristic that mimics the process of natural selection [90]. This heuristic is commonly used to generate useful solutions to optimization and search problems. However, this method has never been adopted for controlling offshore cranes to the best of our knowledge. In order to automatically approximate the mapping equations for manipulator control, (2.9) and (2.10), a continuous GA is employed. The forward kinematic model is all that this approach requires. The set-up of the proposed algorithm is the same, independently of both the manipulator model being controlled and the selected control mode (position or velocity). Furthermore, when selecting the control mode and controlling the manipulator, the same GA instance is used; the only difference lies in the semantics and the size of inputs and outputs.

The proposed algorithm's flow chart is shown in Figure 3.6. The procedure is iterative and at each iteration, a particular cost function is used to move a population of candidate solutions or chromosomes toward better solutions. The key steps of the algorithm are described below.

Define genetic representation, cost function and target cost. Initially, the genetic representation, the cost function and the target cost are defined. In particular, each chromosome encodes its own properties or genes that consist of a set of joint angles, θ_g, constrained within their corresponding limits. The length of each chromosome is equal to the number of joints to be controlled.

The fitness of every individual in the population is evaluated by using a cost function that assesses the Euclidean distance between the target position, $\mathbf{x}_t$, and the actual position, $\mathbf{x}_a$:

$$d(\mathbf{x}_t, \mathbf{x}_a) = |\mathbf{x}_t - \mathbf{x}_a|, \tag{3.3}$$

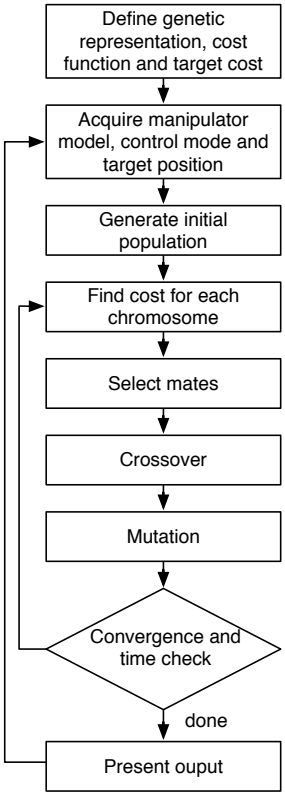


Figure 3.6: Flow chart of the proposed mapping method based on GAs.

where, the actual position is calculated by using forward kinematics, while the target position depends on the input and it is given by:

$$\mathbf{x}_t = \mathbf{x}_{ds}, \tag{3.4}$$

if operating in position control mode, or by:

$$\mathbf{x}_t = \mathbf{x}_a + \dot{\mathbf{x}}_{ds}\Delta t, \tag{3.5}$$

if operating in velocity control mode, where Δt is the time interval between two successive iterations.

Acquire manipulator model, control mode and target position. The main iteration loop starts acquiring the proper manipulator model and the control mode according to the operation scenario. Moreover, the corresponding target position is normalised according to the workspace of the manipulator to be controlled. This will result in relating the cost function to the corresponding workspace.

Generate initial population. Subsequently, an initial population is randomly generated, consisting of 50 individuals for each DOF of the model to be controlled. This choice is empirically determined.

Find cost for each chromosome. The evolution process, which is a sub-loop of the main loop iteration, starts from the initial randomly-generated population and, at each generation, the fitness of every chromosome is evaluated according to the previously defined cost function (3.3). Additionally, any individual with genes that violate the corresponding joint limits gets its cost increased by a considerable factor. Then the modification process of each individual's genome starts in order to form a new generation.

Select mates. The *stochastic universal sampling method* [91], which is a fitness proportionate selection method, is used to select candidates that will be used as parents in the crossover process.

Crossover. The crossover function is defined as a single-point and uniform crossover method with a 50% crossover probability. This is used to create new offspring from the selected parent chromosomes.

Mutation. Mutation is taken into account with the random stochastic addition of $\pm 5\%$ to the value of a chromosome's genes. In particular, the chance of mutation is 20% for each gene and a form of elitism is also used, such that 20% of the fittest chromosomes survive unaltered from one generation to the next. Note that since the operator executes

continuous movements during manipulator operation, this form of elitism for survival between sequential iterations and consecutive target positions makes sense: consecutive input vectors do not differ much stochastically in terms of magnitude and direction.

Convergence and time check. The evolution process is repeated until a termination condition is reached. In particular, the fittest chromosome is returned when the cost drops below the predefined target cost, or when the overall population evolution time exceeds 20 ms. At this point the GA stops evolving. Note that the normalisation of the target position according to the workspace of the manipulator relates the cost function to the manipulator. As such, a correlation will be present between the target cost and the considered model. The predefined target cost is weighted and proportionate to each specific workspace. A 20 ms time limit allows for an acceptable level of evolution in the first few iterations, without affecting the operator's perception. For a small number of DOFs, the time limit is seldom reached stochastically for target positions located inside the workspace boundaries after the first few iterations.

Present output. The genes of the fittest chromosome are then presented as output. In particular, denoting these genes as θ_f and according to the operation scenario, the output is obtained as:

$$\theta_d = \theta_f, \tag{3.6}$$

when operating in position control mode, or as:

$$\dot{\theta}_d = \frac{\theta_f - \theta_a}{\Delta t}, \tag{3.7}$$

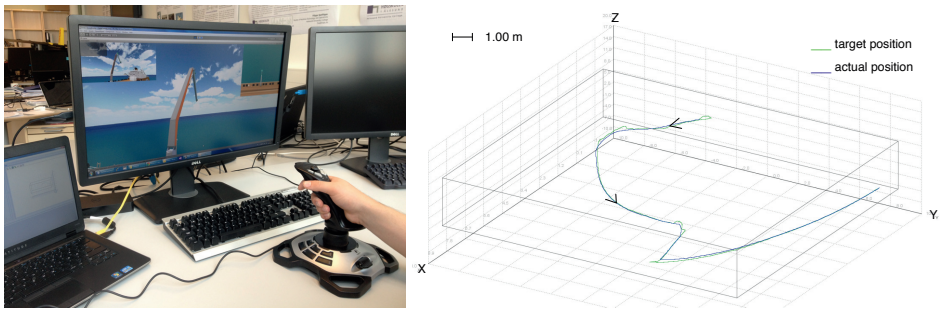
when operating in velocity control mode.

Note that, in this specific case study, the control of the end-effector's orientation is not considered as part of the mapping algorithm but it can easily be included.

Related simulations and results are presented in the following.

3.2.1.1 Simulation Results

Related simulations of the proposed alternative control method based on the use of GAs are carried out. A joystick from *Logitech*, the *Extreme 3d pro* is used as a universal input device on the client side. Each degree of freedom of the joystick corresponds to a translational axis in the workspace of the crane to be controlled. When operating in position control mode, the joystick works as a position proportional replica whose motion maps exactly to the motion of the crane end-effector with constant speed, while, when operating in velocity control mode, a movement of the joystick in a particular direction will produce a translational motion in the same direction at a velocity proportional to



(a) (b)

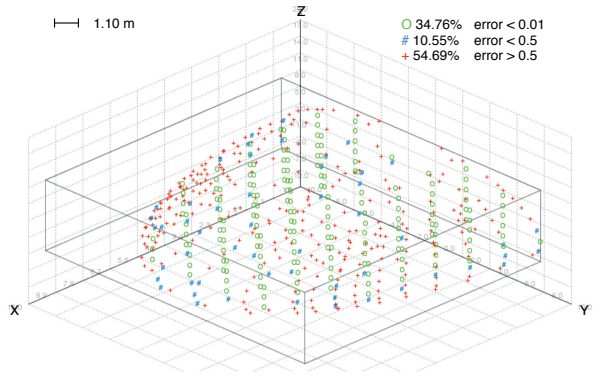
Figure 3.7: The simulation of (a) the knuckle boom crane model and (b) the trajectory tracking of its Cartesian paths in X, Y and Z coordinates.

the joystick displacement. In both cases, when the operator's hand is removed from the joystick, the latter returns to automatically its starting point. Note that, thanks to the modularity of our control architecture, any other joystick or input device can be used without affecting the effectiveness of the system.

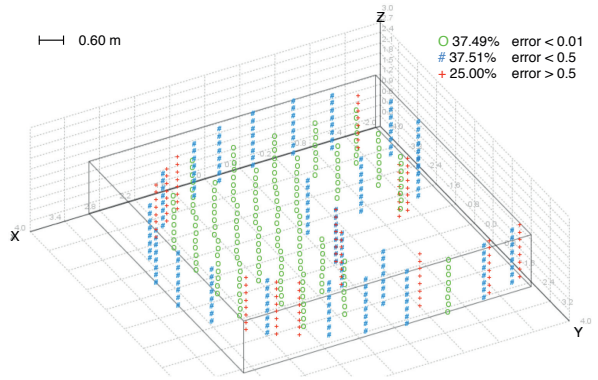
The proposed control method based on the use of GAs is used to control different manipulator models. In detail, a knuckle boom crane, which is shown in Figure 3.7-a, a *SCARA* robot and a *KUKA youBot* robot are modelled and simulated. Regarding the visualisation environment, the game engine *Unity3D* [81] is used to visualise the different models. However, any other visualisation environment could be used without affecting the effectiveness of the proposed architecture.

For each of these models, a trajectory tracking analysis of the Cartesian paths for X, Y and Z coordinates is performed and the result for the knuckle boom crane model is shown in Figure 3.7-b. Generally, the proposed system has demonstrated a quite fast reaction to the inputs. The average response time rarely exceeds the 20 ms time limit. However, this kind of test is task-dependent.

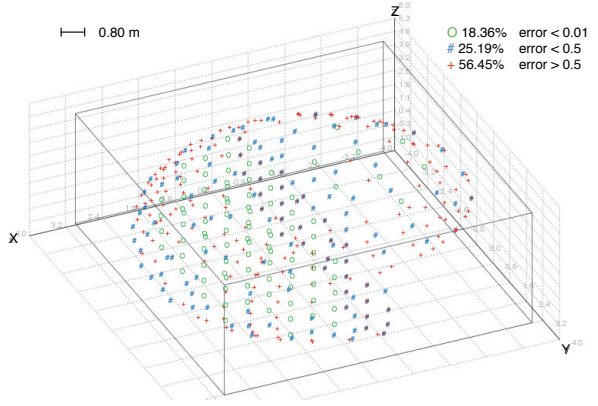
In addition, for each of these models, an error distribution analysis is performed considering a set of 512 equally-spaced target positions in the volume box that encloses their corresponding workspace. For the three models considered, these error distributions are shown in Figure 3.8 as 3D scatter plots. It is probable that the target positions at the boundaries of this imaginary box are less reachable by the manipulators due to their joint constraints, which is why the corresponding errors are stochastically greater. However, even for these points, the proposed control method is able to find the closest position match, thus avoiding potential singularity problems.



(a)



(b)



(c)

Figure 3.8: 3D Scatter plots showing the error distribution (in percentage) for 512 equally-spaced target positions in the volume box that encloses the workspace of (a) the knuckle boom crane model, (b) the *SCARA* robot and (c) the *KUKA youBot* robot.

3.2.2 An Alternative Control Algorithm Based on PSO

PSO is a stochastic global optimisation method that optimises a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality (fitness or cost) [39]. A population of candidate solutions, dubbed particles, is used to solve the problem. These particles are moved around the search-space according to simple mathematical rules defining the particle's position and velocity. Each particle's movement is influenced by its local best known position but is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions. This method has been widely used in robotics [92]. However, this method has never been adopted for controlling offshore cranes to the best of our knowledge.

Using PSO, our system automatically infer the mapping function, (2.10), for the different manipulators to be controlled. This approach only requires the FK models. Note that the unique feature of this method compared to previous works is that the same set-up of the proposed algorithm is adopted independently of the manipulator being controlled and whether the selected control mode is position or velocity. Moreover, when controlling each specific manipulator and once selecting the particular control mode, the same instance of PSO is continuously used; what differs are the semantics and the size of inputs and outputs which are dynamically and automatically set by the system. The algorithm flow chart is shown in Figure 3.9. In the following, the key steps of the algorithm are described.

Define particles representation, cost function and target cost. Each particle consists of a set of joint angles, θ_p, constrained within their corresponding limits. The size of each particle is equal to the number of joints to be controlled. The quality of every particle in the population is evaluated using a composed cost function that assesses two different contributions: the end-effector displacement error, a, and the joint configuration error, b. The first contribution is the Euclidean distance between the target position, $\mathbf{x}_t$, and the actual position, $\mathbf{x}_a$:

$$a = d(\mathbf{x}_t, \mathbf{x}_a) = |\mathbf{x}_t - \mathbf{x}_a|, \tag{3.8}$$

where the actual position is calculated by using forward kinematics, while the target position depends on the input and it is given by:

$$\mathbf{x}_t = \mathbf{x}_{ds}, \tag{3.9}$$

if the system is operating in position control mode, or by:

$$\mathbf{x}_t = \mathbf{x}_a + \dot{\mathbf{x}}_{ds}\Delta t, \tag{3.10}$$

if operating in velocity control mode, where Δt is the time interval between two subsequent iterations. The second component of the proposed cost function considers the joint

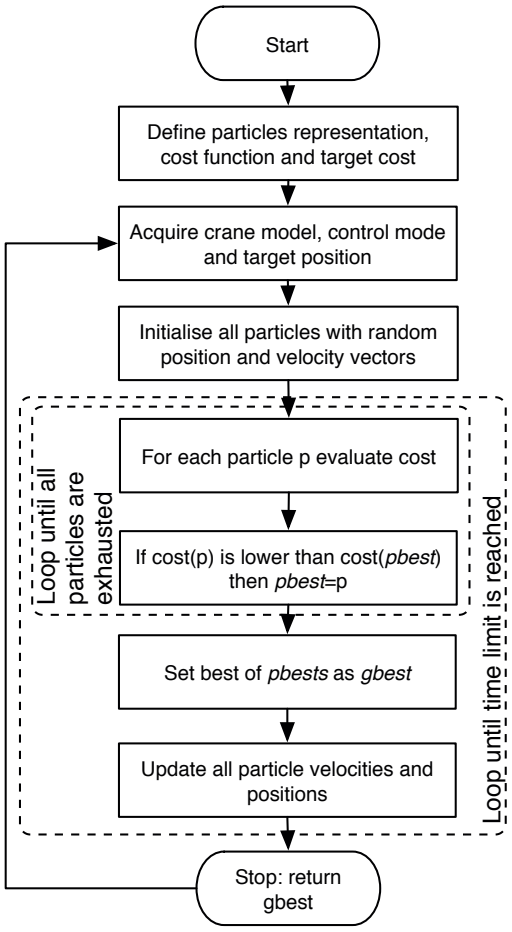


Figure 3.9: The flow chart of the proposed control method based on PSO.

configuration error and it is calculated as the absolute difference between the candidate joints configuration, θ_c, and the actual joint configuration, θ_a:

$$b = |\theta_c - \theta_a|, \tag{3.11}$$

where the actual joint configuration is calculated by using forward kinematics. This contribution is taken into account in order to avoid singularity problems when considering redundant manipulator models. The complete cost function is calculated as:

$$cost = \alpha a + \beta b, \tag{3.12}$$

where $\alpha, \beta \in \Re^+$ are weighting factors such that $\alpha + \beta = 1$. In this specific case, the following values have been selected: $\alpha = 0.8$ and $\beta = 0.2$.

Acquire manipulator model, control mode and target position. The main iteration loop acquires the correct manipulator model and control mode according to the operation scenario. According to the manipulator's workspace, the corresponding target position is normalised. This makes it possible to establish a correlation between the cost function and the corresponding workspace.

Initialise all particles. An initial population of particles is randomly generated. The population size is empirically defined as 10 times the number of DOFs present in the model to be controlled. All particles are initialised with random position and velocity vectors.

Evaluate the cost for each particle. The optimisation process starts with the initial randomly initialised population. This is a sub-loop of the main iteration and at each step, the defined cost function is used to evaluate the cost of every particle. Each particle tracks its coordinates associated with the optimal solution achieved thus far in the solution space. This value is called personal best (*pbest*). The global best (*gbest*) is another value that is tracked by the PSO. It is the best value obtained thus far by any particle in its respective neighbourhood. Following this, the modification process of each particle's velocity and position begins.

Update particle velocities and positions. According to [93], each particle's velocity at iteration $k+1$ can be modelled according to:

$$\mathbf{v}(k+1) = w\mathbf{v}(k) + c_1\mathbf{R}_1(\mathbf{pbest} - \mathbf{s}(k)) + c_2\mathbf{R}_2(\mathbf{gbest} - \mathbf{s}(k)), \tag{3.13}$$

where w is an inertia weight that weighs the contribution of the previous velocity to control the particle's momentum, $\mathbf{v}(k)$, is the velocity of the particle at the iteration k, $\mathbf{s}(k)$ is the current searching point, $\mathbf{R}_1$ and $\mathbf{R}_2$ are vectors that are the same size of the swarm population and contain random numbers in the range of $[0,1]$, c_1 is a learning factor

that determines the importance of *pbest*, and c_2 is a learning factor that determines the importance of *gbest*. c_1 and c_2 are chosen through a higher level optimisation process.

Following this, modification of each particle's next position is obtained by adding the position at iteration k to the distance the particle will travel with the new velocity $\mathbf{v}(k+1)$:

$$\mathbf{s}(k+1) = \mathbf{s}(k) + \mathbf{v}(k+1). \tag{3.14}$$

Convergence and time check. The optimisation process is repeated until a termination condition is satisfied. In particular, when the cost drops below 0.01, or when the overall time spent updating the population exceeds 20 ms, the PSO stops updating and the global best particle is returned. Note that since the target position is normalised according to the workspace of the manipulator to be controlled, a correlation will exist between the target cost and the model considered, as the cost function is related to the manipulator. In this way, the predefined target cost is weighted and proportionate to each specific workspace. Given the 20 ms population updating time limit, the population is able to reach an acceptable level of fitness in the first few iterations without affecting the operator experience. For a small number of DOFs, After the first few iterations, the time limit is rarely reached for target positions located within the workspace boundaries.

Present Output. According to the operational scenario, the output is obtained by:

$$\theta_d = \theta_{pbest}, \tag{3.15}$$

when operating in position control mode, or with:

$$\dot{\theta}_d = \frac{\theta_{pbest} - \theta_a}{\Delta t}, \tag{3.16}$$

when operating in velocity control mode.

Related simulations and results are presented in the following. In addition an extensive comparison between the two presented control methods for controlling maritime cranes and robotic arms is presented in Chapter 5.

3.2.2.1 Simulation Results

Related simulations of the proposed alternative control method based on the use of PSO are carried out. In this case, a *Microsoft Xbox 360* joystick controller is used as a universal input device on the client side. This choice is motivated by the fact that this controller is very user friendly. Also in this case, the chosen input controller works as a translational replica for the axis in the workspace of the crane to be controlled.

A knuckle boom crane with 3 DOFs is modelled and simulated. See Table 3.3 for the crane's D-H table. In this specific case, the proposed mapping method is performed on an

Table 3.3: D-H table of the knuckle boom crane, where $L_1 = 2.62m$, $L_2 = 7.01m$ and $L_3 = 3.46m$.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	L_1	θ_1
2	$\frac{\pi}{2}$	0	0	θ_2
3	0	L_2	0	θ_3
4	0	L_3	0	0

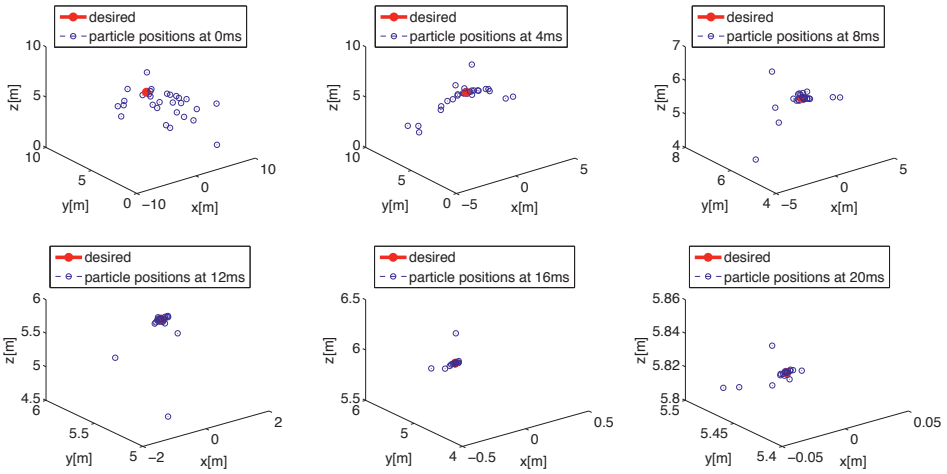


Figure 3.10: The particle updating process to reach a desired target position is shown for an iteration time of 20ms. Starting from the top left corner of Figure 3.10, 6 subsequent screenshots are taken during the considered iteration (at 0ms, at 4ms, at 8ms, at 12ms, at 16ms and at 20ms). It should be noted that the scale of representation changes such that the plots are zoomed-in as the particles get closer and closer to the target.

Intel Core i7-3820QM machine. In Figure 3.10, the particles updating process to reach a desired target position is shown for an iteration time of 20ms. Starting from the top left corner of Figure 3.10, 6 successive time-plots are taken within the same considered iteration (at 0ms, at 4ms, at 8ms, at 12ms, at 16ms and at 20ms). Step by step, the particles get closer and closer to the target. It should be noted that the scale of representation changes so that the plots are zoomed-in as the particles get closer and closer to the target.

Chapter Summary

In this chapter, we presented several alternative algorithms that are able to scale and control different manipulators regardless of inequalities in size, kinematic structure, DOFs,

body morphology, constraints and affordances.

With regard to modular robotic hands, our results are summarised in the following.

- Despite the simplicity of a modular manipulator model, with the increase in the number of its fingers and modules, it becomes rival to the human hand in terms of complexity. To deal with this challenge, a biologically-inspired synergistic control method was adopted in this chapter. This method is inspired to the human hand not only with regard to size and configuration, but also as regards to the control. We implemented the adopted method by using *ModGrasp*, our virtual and physical rapid-prototyping framework for modular robotic hands which was previously presented in Chapter 2.
- As a case study of the adopted control algorithm, a mind-controlled, low-cost modular manipulator was presented. Related simulations were carried out in order to test the presented synergistic control method within the particular case study.

Similarly, the possibility of developing alternative control methods is also investigated concerning the field of maritime cranes and robotic arms. Our findings are listed in the following.

- The inverse kinematics may have multiple solutions when considering arms with redundant DOFs and singularity problems could arise. Moreover, classic kinematic approaches could hardily be applied when considering to control different manipulators using a universal input device because several IK models would be needed: one for each arm or crane to be controlled. In this chapter, we have considered an alternative strategy, which consist of using methods that derives the kinematic properties by applying optimisation procedures. In this way, the system is able to automatically obtain the kinematic properties of different arms, and new models can also easily be added.
- In this regard, two alternative control methods were presented in this chapter aiming to transparently control different manipulators and configurations. The first method is based on the use of GAs. The second method involves the use of PSO. Both the proposed control methods are implemented based on the flexible control architecture for maritime cranes and robot which was previously presented in Chapter 2. Related simulations have been carried out to prove the effectiveness of the two proposed control methods.

CHAPTER 4

System Integration

In this chapter, the challenging problem of system integration is addressed.

With regard to modular robotic hands, the integration between virtual and real prototypes is considered. In particular, more details about the integration of real modular manipulators with the previously introduced rapid-prototyping framework of *ModGrasp* are presented. This system integration allows for establishing a real-time one-to-one correspondence between virtual and physical prototypes.

Similarly, the system integration challenge is successively investigated in the field of maritime cranes and robotic arms. In detail, the integration of our previously introduced flexible control architecture for maritime cranes and robots with a simulation environment specifically designed for offshore applications is presented. The considered simulation environment is the Crane Simulator from the Offshore Simulation Centre AS (OSC) [57].

Next, the system integration and control of real industrial robotic arms is studied. Restricting the focus to *Kuka* industrial robots, we present *JOpenShowVar*, an open-source cross-platform communication interface to *Kuka* robots that allows for reading and writing variables and data structures of the controlled manipulators. Four case studies are presented to demonstrate the potential of *JOpenShowVar*. The first two case studies are open-loop applications, while the last two case studies describe the possibility of implementing closed-loop applications. In the first case study, the proposed interface is used to make it possible for an *Android* mobile device to control a *Kuka KR 6 R900 SIXX (KR AGILUS)* manipulator. In the second case study, the same *Kuka* robot is used to perform a two-dimensional line-following task that can be used for applications like advanced welding operations and similar. In the third case study, a closed-loop application is developed to control the same manipulator with a *Leap Motion Controller* that supports hand and finger motions as input without requiring contact or touching. In the fourth case study, a bidirectional closed-loop coupling is established between a *Force Dimension omega.7* haptic device and the same *Kuka* manipulator. Related experiments are carried out to validate the efficiency and flexibility of the proposed communication interface.

Finally, the integration of our cranes and robots control system (presented in Chapter 2) with a physical motion platform, which simulate the wave effects, is considered. The system consists of an industrial robotic arm and a motion platform with three degrees of

freedom. The motion platform is used to simulate the wave effects, while the robotic arm is controlled by the user with a joystick.

Contributions of this chapter: referring to modular robotic hands, we contribute to this chapter by analysing the integration between virtual and real prototypes. We present several implementation details about how this can be realized with *ModGrasp*, our previously presented rapid-prototyping framework. This system integration is a relevant contribution for the thesis because it allows for conducting experiments with both virtual and real modular prototypes in a transparent fashion. Different application scenarios are possible.

With regard to the field of maritime cranes and robotic arms, the flexible control architecture for maritime cranes and robots presented in Chapter 2 is integrated with the Crane Simulator from the Offshore Simulation Centre AS (OSC) [57]. This integration is important because because it makes it possible to establish a research tool for the investigation of alternative control algorithms, which can be efficiently tested in a realistic maritime simulation environment.

Successively, we contribute to the system integration and control of real industrial robotic arms. In particular, we introduce *JOpenShowVar*, an open-source cross-platform communication interface to *Kuka* industrial robots. *JOpenShowVar* is an important contribution because it opens up to a variety of possible applications making it possible to use different input devices, sensors and to develop alternative control methods. This is relevant from both a research as well as an application point of view.

The last contribution of this chapter concerns the integration of our cranes and robots control system with a physical motion platform. This system integration is important because it gives researchers the possibility of testing alternative control algorithms for maritime cranes and robotic arms in a realistic and safe laboratory setup.

Organization of this chapter: Following the same line of the thesis, this chapter is divided in two main sections. In Section 4.1, more details about the integration between virtual and physical modular robotic hands are discussed. Similarly, in Section 4.2, we first present the integration of our previously introduced flexible control architecture for maritime cranes and robots with the Crane Simulator from the Offshore Simulation Centre AS (OSC). Then, the system integration and control of real industrial robotic arms is studied, restricting the focus to *Kuka* industrial robots. Finally, the integration of our cranes and robots control system with a physical motion platform, which simulate the wave effects, is considered.

Publications: The results of this chapter concerning modular robotic hands are based on the papers [62] and [63]. The findings related to maritime cranes and robotic arms are based on the papers [83], [94], [95] and [96].

4.1 Integration in the Field of Modular Robotic Hands

Considering the field of modular robotic hands, more details about integration of real modular devices with the previously introduced rapid-prototyping framework of *ModGrasp* are presented. A real-time one-to-one correspondence between virtual and physical prototypes is established thanks to this system integration.

4.1.1 Integration Between Virtual and Real Modular Robotic Hands

ModGrasp, our open-source rapid-prototyping framework for designing low-cost sensorised modular hands was introduced in Chapter 2. More implementation details are presented in this section focusing on the system integration with real modular prototypes. Each module of the real prototypes is actuated by using one *HiTEC HS-85MG* micro servo. This particular micro servo features a torque constant $K_t = 0.50Nm/A$ and a maximum torque of $0.294Nm$.

Wiring Schematic. The master-slave communication pattern that is provided with *ModGrasp* allows for an transparent system integration between virtual and real models. A section of the wiring schematic for the hardware is shown in Figure 4.1. As already mentioned, an *Arduino Uno* board [77] based on the *ATmega328* micro-controller is used as the master, while one *Arduino Nano* [77] board is used as a slave to control each module. The master is connected to a Bluetooth module, which enables the communication with external input devices. For instance an EEG headset can be used to control the modular models, as shown in Chapter 3.

Low-cost Torque Sensing Implementation. In order to monitor the load of each joint actuator, the current is continuously measured from each slave controller. To measure the current load, two 1Ω resistors connected in parallel are used for each servo. These resistors are connected from the servo ground to the common ground, such that the current will run through that point. Since the electrical resistance is very small, the current can be read as the change in voltage over these resistors. Therefore, wires are connected from these resistors to the analogue input on the respective slave controller board.

Communication Details. To implement the I^2C protocol, which allows for communication between the master and the slaves, two analogue inputs, the A_4 and A_5, are connected together for all the controller boards. Additionally, a $1.5k\Omega$ resistor is placed between A_4 and $5V$ and another is placed between A_5 and $5V$ to ensure that the *Arduino* boards do not interpret noise as an actual high or low value. Two switches are also added in order to make it easy to turn on the power to the master and the slaves.

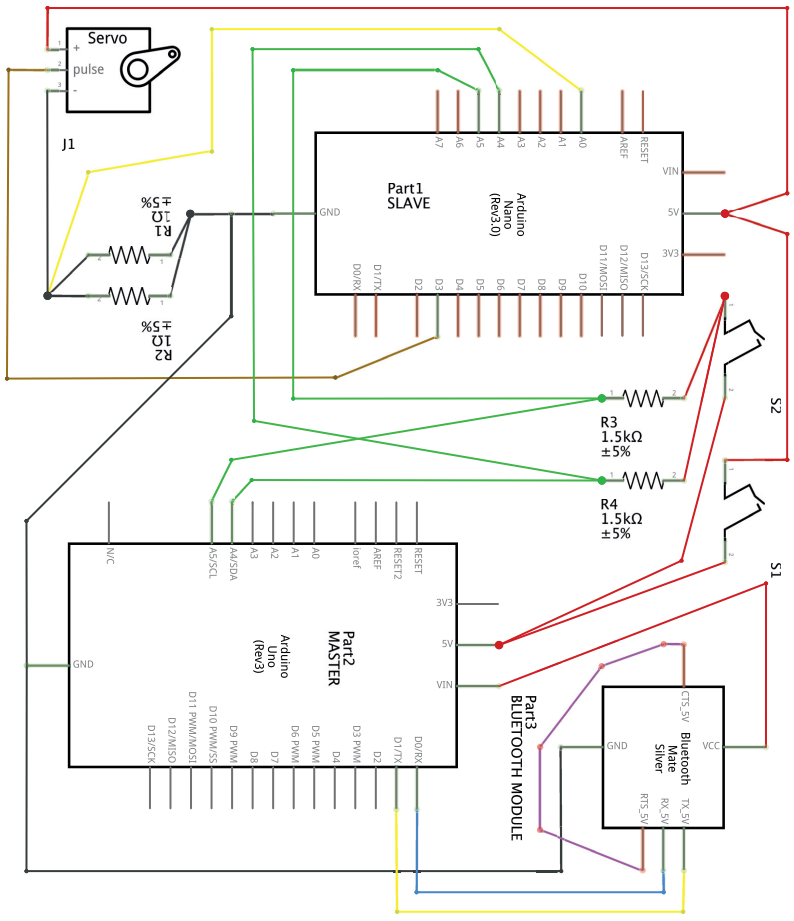


Figure 4.1: A section of the wiring schematics for the low-cost sensing circuit and for the communication between master and slave. The master is connected to a Bluetooth module to enable the communication to a EEG headset or other possible input devices.

Libraries. An implementation of the I^2C protocol provided by the *EasyTransfer* library [97] is used for a fast and easy implementation. The library, besides offering a good support for the I^2C protocol, also provides some extra interfaces and features a wide range of data types that can be easily transferred. Moreover, the *RXTX Java library* [98] is employed to implement the communication link between the simulation environment and the physical manipulator prototypes.

This system integration allows for establishing a real-time one-to-one correspondence between virtual and physical prototypes. Different configurations and various control methods can be transparently tested.

4.2 Integration in the Field of Maritime Cranes and Robotic Arms

In the previous sections, the challenging problem of system integration has been addressed with regard to modular robotic hands. In this section, this same issue is similarly considered for the field of maritime cranes and robotic arms. In particular, the integration of our previously introduced flexible control architecture for maritime cranes and robots with the Crane Simulator from the Offshore Simulation Centre AS (OSC) is first presented. Then, the system integration and control of real industrial robotic arms is studied with particular focus to *Kuka* robots. We present *JOpenShowVar*, an open-source cross-platform communication interface to *Kuka* industrial robots that allows for reading and writing variables and data structures of the controlled manipulators. Finally, the integration of our cranes and robots control system with a physical motion platform, which simulate the wave effects, is considered.

4.2.1 Integrated Flexible Maritime Crane Architecture for the Offshore Simulation Centre AS (OSC)

Maritime cranes are usually operated in a very complex and demanding environment. The control operation involves many problems such as load sway, positioning accuracy, wave motion compensation and collision avoidance. Therefore, the safe, accurate and efficient operation of these kinds of cranes is challenging. In addition, testing new control methods in a real set-up environment is very difficult because of the challenging operational workspace of maritime cranes. Due to the challenging operational crane scenarios in real applications, a promising approach consists of using a computer-simulated environment.

To overcome the low control flexibility and non-standardisation issues that affect the current maritime crane control system, a flexible control architecture that allows for implementing and testing different control methods in a computer-simulated environment was previously presented in Chapter 2. Since the presented architecture is implemented in

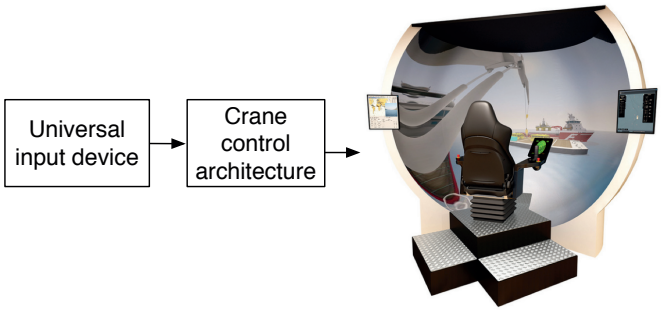


Figure 4.2: The idea of integrating our flexible control architecture with the Crane Simulator from the Offshore Simulation Centre AS (OSC).

Java, a portable and general-purpose programming language, a variety of different possibilities are available, including the use of different third-party libraries and visualisation environments. Regarding the visualisation environment, in our preliminary work, the game engine *Unity3D* [81] was used to visualise the different models. However, even though *Unity3D* provides a set of advanced 3D graphics tools, it is mainly a game engine and does not inherently provide specific support for maritime applications such as wind or wave generation. This justifies the need for switching to a simulation environment specifically designed for offshore applications.

Few examples of such dedicated and sophisticated simulation environments exist in maritime applications [99]. Moreover, little work has been done concerning the standardisation process [100], [101]. The Offshore Simulator Centre AS (OSC) [57] is an advanced provider of simulators for demanding naval operations. The centre benefits from the high level of operational know-how of the maritime cluster in Norway combined with advanced computer technology, sophisticated mathematical models and state of the art 3D graphics display systems. The OSC offers simulation of complete offshore vessel bridges for which all relevant controls and systems can be implemented. During each training session, it is possible to change the vessel's environment, including altering the weather, winds, waves and time of day at the touch of a button by using intelligent software and interfaces. However, the main objective of such an advanced simulator is to provide a realistic training environment. The OSC does not naturally offer any flexible methods concerning the control methodology. In fact, each crane model is controlled by using a dedicated control algorithm that is not possible to modify or access at run-time. As a result, it is not possible to switch between different control methods at run-time, nor is it possible to investigate alternative control approaches.

We decided to integrate our flexible and reliable control architecture with the OSC crane simulator. The underlying idea of this system integration is shown in Figure 4.2. The Google *Protocol Buffers* protocol [102] is adopted to realise the communication protocol. By achieving this architectural coupling, a base for alternative maritime crane control algorithm research is created. Different approaches can be efficiently tested in a realistic

maritime simulation environment. It is possible to dynamically switch between different methods and models at run-time. These new possibilities offer several advantages. From a research point of view, more efficient and safe control techniques can be investigated, while, from a training prospective, different realistic operation scenarios can be experienced by the crane operators. In the following, the key elements of the proposed system integration are presented.

The OSC Crane Simulator. The OSC Crane Simulator is designed for optimal training and education of crane operation personnel. An interior view of the OSC Crane Simulator is shown in Figure 4.3. Based on advanced simulation and visualisation technology, the simulator can be configured in a number of ways, regarding hardware set-up, display solution, and software setup (including the type of crane, types of vessels/lifting objects and training scenarios). The main components of the OSC Crane Simulator are:

- OSC Offshore Simulator Software. It includes all the crane simulation functionality;
- Instructor Station. It is an optional component that allows the instructor to supervise the training operations;
- Deck Personnel functionality. It allows for including personnel such as a banksman, a slingerman or a signalman;
- Training scenarios and cases. It is possible to select different cases and operation scenarios;
- Crane Simulator hardware. The hardware set-up can be customised according to the desired need;
- Virtual World and 3D Graphics. The simulated environment can be customised according to the desired application;
- Crane driver chair with armrest, joysticks, buttons and touch screens;
- Display Solution options. Different solutions can be used for visualisation.

In addition, depending on available space, the solution can be configured with the Crane Cabin inside a “dome” (like the one shown in Figure 4.2), with only a crane operator chair in front of a smaller “dome” or with LCD screens. The most realistic set-up uses an array of projectors to visualise the simulated environment on the inner surface of the “dome”. Advanced software algorithms are used to align the different components of the scene and to avoid any kind of distortion.

Together with Aalesund University College, the OSC has implemented a highly regarded and sophisticated team training concept for offshore crew, both for Crane Operations, Anchor Handling and Platform Supply Vessel operations. This allows the entire team to be trained together in rigorous real-life exercises. This multiple training concept gives

Figure 4.3: An interior view of the OSC Crane Simulator (courtesy of Offshore Simulator Centre AS).

all team members the advantage of working together to avoid operational mistakes, misunderstandings and to increase safety on the real job. This way of learning builds very effective teams. It should be noted that since the OSC Crane Simulator is a commercial product, internal implementation details cannot be provided.

Even though, the OSC Crane Simulator provides such advanced simulation tools, the main drawback is that it is designed for realistic training purposes, stressing the focus on usability and user experience. Because of this, the OSC does not offer any flexible methods concerning the control methodology. In particular, each crane model is controlled by using a dedicated control algorithm that is not possible to modify or access at run-time. As a result, it is not possible to dynamically switch between different control methods, nor is possible to easily investigate alternative control approaches. This motivates the integration with our flexible architecture.

Architecture Integration. The integrated control system architecture is shown in Figure 4.4. It is a client-server architecture with the input device running as a client and communicating with a server where the logic of the control algorithm is implemented. The controlled arms are simulated in the OSC crane simulator, which also acts as a client and provides the user with an intuitive visual feedback. The proposed architecture provides the possibility of controlling the arms in position mode or velocity mode. When the operator manoeuvres the manipulator, the movements are transferred from the OSC Simulator to our control framework. A vector signal with no semantic, $\mathbf{s}$, is sent from the universal input device to the server. Here, according to the operational scenario, the vector signal is interpreted as the desired position $\mathbf{x}_d$ or the desired velocity vector $\dot{\mathbf{x}}_d$. The information flow and the logic of the control system have been already introduced in Chapter 2. The calculated desired joint angles θ_d or joint velocities $\dot{\theta}_d$ are then forwarded from the server to the OSC crane simulator in order to actuate the crane model. As feedback from the OSC crane simulator, the actual joint angles θ_a and joint velocities $\dot{\theta}_a$ are

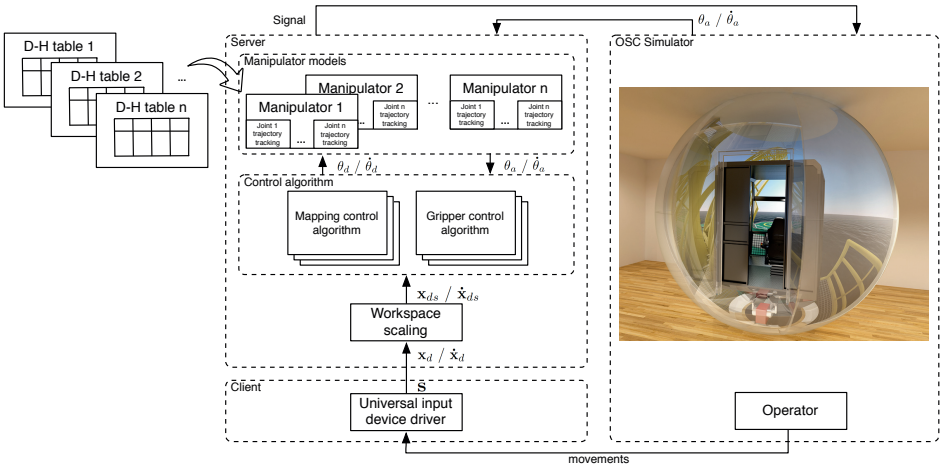


Figure 4.4: The integration of our flexible control architecture with the OSC Crane Simulator.

sent back to the server and can be used by the control mapping algorithm.

Communication Protocol. In this section, the integration of the presented framework architecture with the OSC Crane Simulator is outlined. For this integration, we have chosen to adopt the Google *Protocol Buffers* protocol [102]. The reason for this choice of protocol is that it is a requirement from the OSC. In fact, the OSC widely uses *Protocol Buffers* for several applications. In addition, there are several advantages that comes from using this specific data interchange format instead of other similar alternatives in terms of flexibility, reliability and performance [103]. Furthermore, *Protocol Buffers* are a robust, efficient and automated mechanism for serialising structured data. This mechanism allows for defining how data should be structured, after which it is possible to use specially generated source code to easily write and read the structured data to and from a variety of data streams and using a variety of languages. It is even possible to update the data structure without breaking deployed programs that are compiled against the "old" format.

In detail, to use *Protocol Buffers*, it is necessary to generate code for each message that needs to be encoded, and use the generated code to encode/decode the message, as shown in Figure 4.5. *Protocol Buffers* use a binary encoding format that allows for specifying a data schema using a specification language. Each protocol buffer message is a small logical record of information, containing a series of name-value pairs. For our integration case, the adopted *.proto* file is shown in the Algorithm 4.1 sketch box. It defines the structure for the messages to be send and to be received. In fact, multiple message types can be defined in a single *.proto* file. The message to be send to the OSC is defined

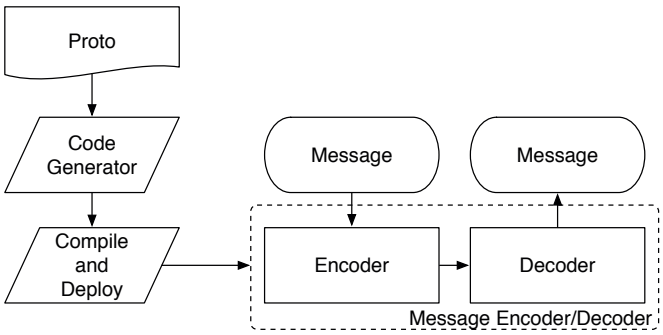


Figure 4.5: To use *Protocol Buffers*, it is necessary to generate code for each message that needs to be encoded, and use the generated code to encode/decode the message.

```
message ToOSC {
required int32 model = 1;
optional string modelName = 2;
repeated double valveSignals = 3 [packed=true];
}

message ToAAUC {
repeated double jointAngles = 1 [packed=true];
repeated double jointVelocities = 2 [packed=true];
}
```

Algorithm 4.1: The *.proto* schema containing the structure of the messages to be send and to be received.

as *ToOSC*. It contains the ID number and the name of the desired crane model to be controlled, as well as the control signal for the crane valves. The ID number for the model is required. Meaning that a well-formed message must have exactly one of this field, while the model name is optional meaning that a well-formed message can have zero or one of this field (but not more than one). The valve signals can be repeated any number of times (including zero) in a well-formed message. The order of the repeated values will be preserved. The reply message to be received by the framework architecture is defined as *ToAAUC*. It contains the actual joint angles and velocities. Both of these variables are marked as repeated fields. In addition, each field in the message definition has a unique numbered tag. These tags are used to identify the fields in the message binary format, and should not be changed once the message type is in use.

Several other benefits of using this approach can be highlighted. Encoding the semantics of the message once, in *.proto* format, is enough to help ensure that the signal does not get lost between applications. Numbered fields in *.proto* definitions obviate the need for version checks which is one of the explicitly stated motivations for the design and implementation of *Protocol Buffers*. The required, optional, and repeated keywords in *Protocol Buffers* definitions are extremely powerful. They allow for encoding, at the schema level, the shape of the data structure, and the implementation details of how classes work are automatically handled.

This system integration establishes the base for the research of alternative control algorithms, which can be efficiently tested in a realistic maritime simulation environment.

4.2.2 Integration and Control of *Kuka* Industrial Manipulators

In the previous sections, the challenging problem of system integration has been addressed with regard to maritime cranes. In this section, this same issue is similarly considered for the field of robotic arms. Only a small number of industrial manipulators with an open control interface has been designed as far as robotics is concerned. Focusing exclusively on *Kuka* industrial robots [27], a cross-platform communication interface that works with all *Kuka* robots has not been released yet to the best of our knowledge. To overcome these problems, *JOpenShowVar*, a Java open-source cross-platform communication interface that allows for reading and writing all the controlled manipulator variables, is presented. This interface allows researchers to use different input devices, sensors and to develop alternative control methods. *JOpenShowVar* library is compatible with all *Kuka* robots that use *KR C*4 or previous versions. The basic concept is shown in Figure 4.6: *JOpenShowVar* works as a *middleware* between the user program and the KRL. Some high-level functions are also provided to enable angles and torques readings of the controlled manipulator. This feedback signal is very important in order to improve the manipulator dexterity and to achieve crucial functions like sensitive collision detection and compliant control actions with closed-loop control. Some guidelines for allowing the user implementing new high-level procedures are discussed. *JOpenShowVar* is an open-source project and it is available on the Internet at `https://github.com/aauc-mechlab/jopenshowvar`,

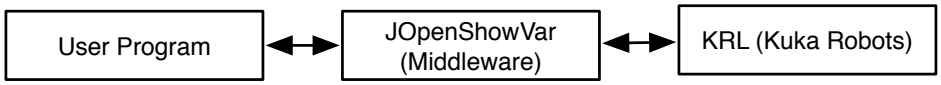


Figure 4.6: The idea of realising a communication interface for *Kuka* robots that works as a *middleware* between the user program and the *Kuka Robot Language* (KRL).

along with several detailed class diagrams, documentation and demo videos.

4.2.2.1 *JOpenShowVar* architecture

The authors initially describe the design choices that characterise the proposed architecture. Successively, the architectural concept is presented, analysing the communication protocol, possible control approaches and some high-level methods.

The design of *JOpenShowVar* is based on the following design choices:

- Low-cost: the developed architecture does not requires any supplementary software packages provided by *Kuka* such as the *Kuka.RobotSensorInterface* [30] or the *Kuka.Ethernet KRL XML* [30]. Therefore, no major capital investments are required to actually purchase these packages from *Kuka*. This fact makes the proposed solution very inexpensive;
- Flexibility: the system offers a virtually unlimited I/O and the possibility of including third party libraries. This allows for adding support for advanced mathematical tools such as matrix operations, optimisation or filtering methods, thus making it very simple to implement novel control approaches;
- Reliability: the system is easy to maintain, modify and expand by adding new components and features. In addition the proposed interface is also open-source and cross-platform;
- Integrability: the proposed system interface presents a modular structure that can facilitate the integration with ROS. Even though this integration is outside the scope of this journal paper, it is considered as an important future work which will surely improve the usefulness of the proposed interface. The community of developers at ROS is looking forward to the integration of JOpenShowVar. The developers have confirmed the usefulness of the proposed interface especially because there are currently no other alternative offering similar features.

Hereafter, several specific functions, variables and configurations related to the KRL and the KRC are referred to in order to introduce the architectural concept. For a more detailed introduction to the KRL, the reader can refer to [28]. The proposed control system architecture is shown in Figure 4.7. It is a client-server architecture with *JOpenShowVar* running as a client on a remote computer and *KUKAVARPROXY* acting as a server on the

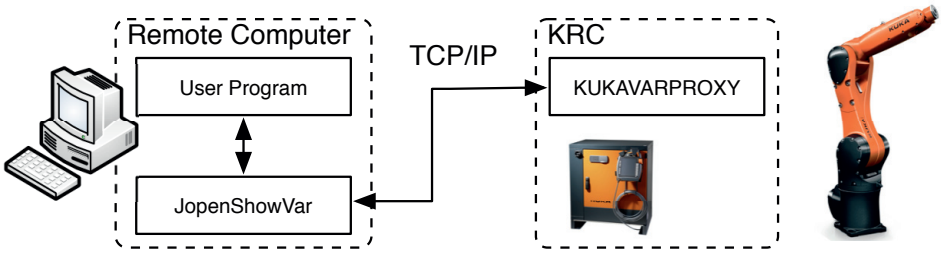


Figure 4.7: The proposed architecture for *JOpenShowVar*: a client-server model is adopted.

KRC. *JOpenShowVar* locally interacts with the user program and remotely communicates with the *KUKAVARPROXY* server via *TCP/IP*.

In particular, *KUKAVARPROXY* is a multi-client server that is written in Visual Basic 6.0 and can serve up to 10 clients simultaneously. *KUKAVARPROXY* implements the *Kuka CrossComm* interface. This interface allows for the interaction with the real-time control process of the robot and makes it possible to perform several operations such as selection or cancellation of a specific program, errors and faults detection, renaming program files, saving programs, resetting I/O drivers, reading variables and writing variables. *KUKAVARPROXY* implements the reading and writing methods. All the variables that need to be accessed by these methods have to be declared as global variables in the predefined global system data list $CONFIG.DAT. All kinds of variables can be declared in this file from basic types such as INT, BOOL and REAL to more complex structures like E6POS and E6AXIS that allow for storing the robot configuration. Moreover, several system variables can be accessed provided there are no restrictions due to the type of data such as for $PRO_IP, $POS_ACT, $AXIS_ACT or $AXIS_INC. For example, the current robot position, $POS_ACT, cannot be written but only read. Restrictions of this nature are checked by the controller.

As already mentioned, the interface of the *Kuka CrossComm* class allows for the interaction with the real-time control process of the robot to be controlled. However, it should be noted that the *Kuka CrossComm* class can only be remotely accessed via *TCP/IP*. Unfortunately, the *TCP/IP* communication introduces inevitable delays, therefore *JOpenShowVar* cannot provide a real-time access to the robot's data. Only soft real-time applications can be realised. In fact, it takes a non-deterministic time to access a specific variable. Since *Kuka* does not offer any kind of documentation on this topic, several experimental tests were performed at our laboratory in order to asses this time interval. According to our experiments, reported later in this chapter, the average access time is about 5 ms. Moreover, this time interval is not affected by the kind of access to be performed (whether it is a reading or a writing operation) or by the length of the message. For these reasons, it is advantageous to aggregate several variables in logical structures when reading or writing data. By using data structures it is possible to simultaneously access several variables,

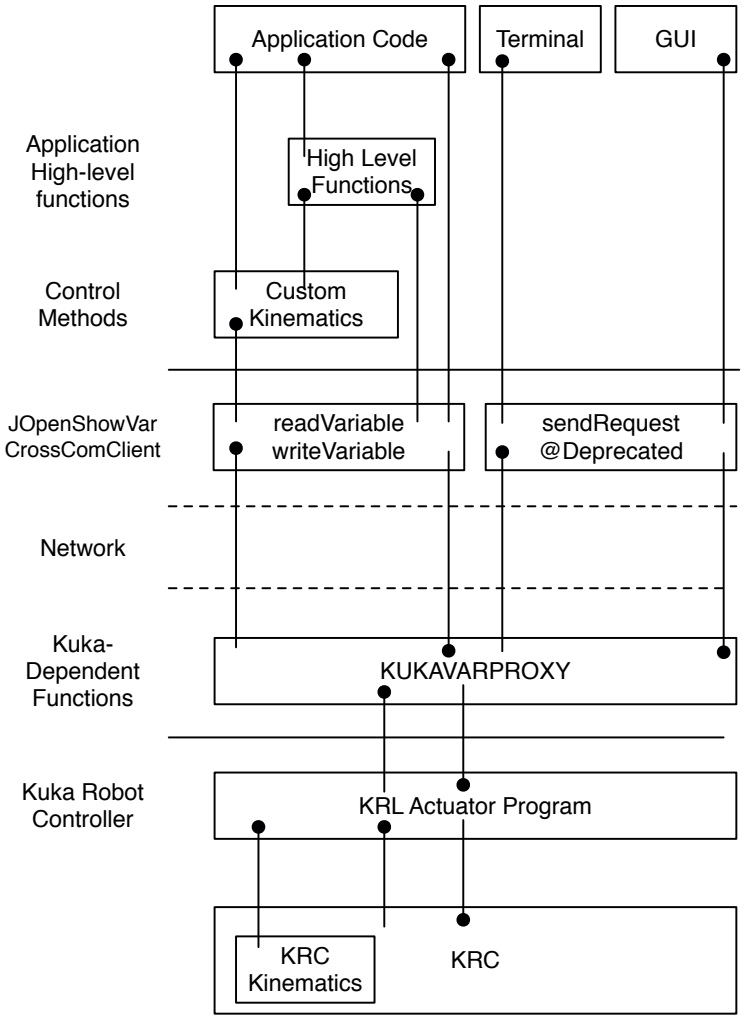


Figure 4.8: The architectural levels of *JOpenShowVar*.

thereby minimising the access time. The only limitation to this approach is on the length of the logical structures that cannot exceed 255 bytes.

JOpenShowVar provides a client, *CrossComClient*, which is written in Java, thus making cross-platform support possible. The architectural details of the *JOpenShowVar* library are shown in Figure 4.8. In our preliminary implementation, the client initially provided only one low level method, *sendRequest*. This method allows for both reading and writing variables. The *sendRequest* method returns a *Callback* instance containing the updated value. However, in the latest release of *JOpenShowVar*, starting from version *v0.2*, the *sendRequest* is marked as a deprecated method, since two new more flexible and efficient methods are introduced: *readVariable* and *writeVariable*.

On top of the *JOpenShowVar*'s methods that implement the low level communication protocol, another logic layer can be added by the user developer allowing for the possibility of implementing alternative control methods (custom kinematics) as well as some higher level functions. The application code can run on top of this hierarchical architecture. In addition a graphical user interface (GUI) and a terminal are provided with *JOpenShowVar* to allow the user for monitoring the robot's state, visualising and manually setting all the desired variables. It should be noted that the GUI still uses the *sendRequest* method of *JOpenShowVar* for a practical reason, since this old method does not require any knowledge on the internal structure of the variables to be accessed compared to the new methods. In the following, the low-level communication protocol, a detailed reference explanation of the newly released methods, the possibility of implementing custom control functions, as well as some guidelines to develop high-level procedures are discussed.

Communication protocol. The communication protocol relies on the TCP/IP protocol. In particular, on top of the TCP/IP layer, specially formatted text strings are used for message exchanges. *KUKAVARPROXY* actively listens on TCP port 7000. Once the connection is established, the server is ready to receive any reading or writing request from the client.

In order to access a variable, the client must specify two parameters in the message: the desired type of function and the variable name.

To read a specific variable, the type of function must be identified by the character "0". For instance, if the variable to be read is the system variable $OV_PRO, which is used to override the speed of the robot, the message that the client has to send to the server will have the following format:

0009007$OV_PRO.

In detail, the first two characters of this string specify the message identifier (ID) with a progressive integer number between 00 and 99. The answer from the server will contain the same ID so that it is always possible to associate the corresponding answer to each request even if the feedback from the server is delayed. The next two characters in the string specify the length of the next segment in hexadecimal units. In this specific case,

Table 4.1: Reading variables.

Field	Description
00	message ID
09	length of the next segment
0	type of desired function
07	length of the next segment
$OV_PRO	Variable to be read

Table 4.2: Writing variables.

Field	Description
00	message ID
0b	length of the next segment
1	type of desired function
09	length of the next segment
$OV_PRO	Variable to be written
50	value to be written

09 accounts for one character specifying the function type, two characters indicating the length of the next segment and seven characters for the variable length. The fifth character 0 in the message represents the type of the desired function, which in this case is reading. Subsequently, there are two more characters indicating the variable length (in hexadecimal units) and finally the variable itself is contained in the last section of the message. Table 4.1 shows a summary of each field of the message with the corresponding description.

To write a specific variable, three parameters must be specified: the type of function, the name of the desired variable and the value to be assigned. The writing function is specified by the character "1". For instance, if the variable to be written is the system variable $OV_PRO with a value of 50 (50% override speed), the message that the client has to send to the server will have the following format:

000b109$OV_PRO50.

Table 4.2 shows a summary of each field of the message with the corresponding description.

Variables, structures and methods. From the release version *v0.2* of *JOpenShowVar*, several new classes have been added to the library to improve the usability. In particular, two abstract classes, *KRLVariable* and *KRLStruct* (which extends *KRLVariable*), are provided to allow the user to implement any KRL variable or structure, respectively. In this way, it is possible to create and maintain a local instance of all the desired variables and structures on the client side. Based on these two abstract classes, the most commonly used KRL variable and structures have been implemented:

- the *KRLBool* class extends *KRLVariable* and represents a Bool variable conform to the KRL standard;
- the *KRLInt* class extends *KRLVariable* and represents an *Int* variable conform to the KRL standard;
- the *KRLReal* class extends *KRLVariable* and represents a *Real* variable conform to the KRL standard;
- the *KRLEnum* class extends *KRLVariable* and represents an *Enum* variable conform to the KRL standard;
- the *KRLAxis* class extends *KRLStruct* and represents an *Axis* struct variable conform to the KRL standard;
- the *KRLE6Axis* class extends *KRLStruct* and represents an *E6Axis* struct variable conform to the KRL standard;
- the *KRLPos* class extends *KRLStruct* and represents a *Pos* struct variable conform to the KRL standard;
- the *KRLE6Pos* class extends *KRLStruct* and represents an *E6Pos* struct variable conform to the KRL standard;
- the *KRLFrame* class extends *KRLStruct* and represents a *Frame* struct variable conform to the KRL standard.

Any other KRL variable or structure that is not included in *JOpenShowVar* library yet can be easily implemented by the user.

Since the release version *v0.2* of *JOpenShowVar*, the *sendRequest* is marked as a deprecated method. To replace this old method, two new, more reliable methods, are added to the *CrossComClient*:

- the *readVariable* method allows for reading any desired remote variable or structure from the controlled robot and store it locally. An exception is thrown if an error in the communication protocol occurs;
- the *writeVariable* method allows for updating any desired remote variable or structure of the controlled robot with the value of the corresponding local variable or structure, respectively. An exception is thrown if an error in the communication protocol occurs.

The deprecated *sendRequest* method is being kept as part of the *JOpenShowVar* library simply because the GUI still uses it for a practical reason. In fact, this old method does not require any knowledge on the internal structure of the variables to be accessed compared to the newly introduced methods. Moreover, it should be noted that the new method *writeVariable* cannot handle arrays; this can only be done by using the old *sendRequest* method. In the Algorithm 4.2 sketch box, a possible use-case example is shown to highlight the differences between the new methods and the deprecated *sendRequest* method.

```
try (CrossComClient client = new CrossComClient("158.38.140.193", 7000)
    ) {
  //JOpenShowVar v0.1 reading
  Callback readRequest = client.sendRequest(new Request(0, "$OV_JOG"));
  System.out.println(readRequest);

  //JOpenShowVar v0.1 writing
  Callback writeRequest = client.sendRequest(new Request(1, "$OV_JOG",
      "100"));
  System.out.println(writeRequest);

  //JOpenShowVar v0.2 reading
  KRLReal jog = KRLVariable.OV_JOG();
  client.readVariable(jog);
  System.out.println(jog);

  //JOpenShowVar v0.2 writing
  jog.setValue(10);
  client.writeVariable(jog);
  System.out.println(jog);
}
```

Algorithm 4.2: A use-case example that highlights the differences between the new methods and the deprecated *sendRequest* method.

Control methods. *JOpenShowVar* opens up to a variety of possible applications making it possible to use different input devices and to develop alternative control methods. In particular, the proposed interface provides the possibility of implementing either a position or a velocity control approach. The user experience is substantially different in each case. When using the position control mode, the operator simply controls the position of the robot's end-effector with constant velocity; when operating in velocity control mode, the operator also sets the velocity of the robot tool. In the first case, when the operator releases the input device, the end-effector moves back to its starting point, while in the second scenario, the arm just stops moving but it keeps the last given position.

To control the robot motion according to the desired operational scenario, *JOpenShowVar* allows researchers to use the standard kinematics provided with the KRC. However, it is also possible to implement alternative control algorithms according to current needs. This is illustrated in Figure 4.9-a and in Figure 4.9-b, respectively. It should be noted that the KRL does not provide a native way to obtain velocity control. When using the KRC kinematics, this limitation can be overcome by expressing the target position as:

$$\mathbf{x}_t = \mathbf{x}_d, \tag{4.1}$$

if operating in position control mode, or by:

$$\mathbf{x}_t = \mathbf{x}_a + \dot{\mathbf{x}}_d \Delta t, \tag{4.2}$$

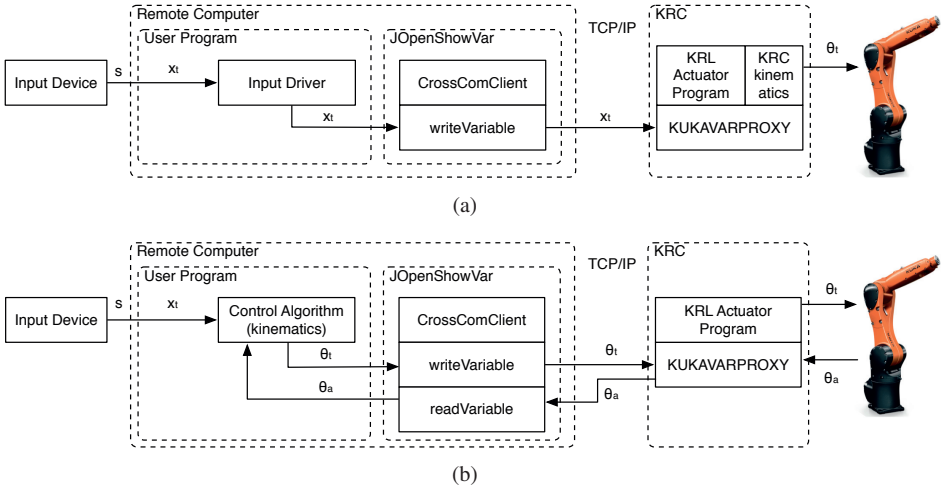


Figure 4.9: (a) the user program utilises *JOpenShowVar* to set the desired end-effector position and then the robot joints are calculated by the KRC using the standard kinematic model, (b) a custom control algorithm can be implemented by the user to calculate the joint values for the robot and then send these angles to the KRC to be actuated.

if operating in velocity control mode, where Δt is the estimated time interval between two successive iterations. As already mentioned, *JOpenShowVar* cannot provide a real-time access to the robot's data. Only soft real-time applications can be realised. It takes a non-deterministic time to access a specific variable. According to our experiments, reported later in this chapter, the average access time is about 5 ms. Therefore Δt can be approximated to a slightly bigger factor of the the average access time. To achieve better performance, the average access time should be continuously monitored and updated. Perhaps, this may be a price to high to pay for some applications with real-time requirements but *JOpenShowVar* still provides great advantages in terms of flexibility.

Alternatively, when a custom control algorithm is needed, the target joint configuration is given by:

$$\theta_t = \theta_d, \tag{4.3}$$

if operating in position control mode, or by:

$$\theta_t = \theta_a + \dot{\theta}_d \Delta t, \tag{4.4}$$

if operating in velocity control mode.

When the operator manoeuvres the manipulator, a vector signal with no semantic, $\mathbf{s}$, is sent from the input device to the user program. Here, according to the operational scenario, the vector signal is interpreted as the target position $\mathbf{x}_t$. If the intent is to use the standard kinematics provided with the KRC, the user program simply works as a driver for the input device and uses the *writeVariable* method of *JOpenShowVar* to forward $\mathbf{x}_t$ to a KRL

```
try (CrossComClient client = new CrossComClient("158.38.140.193", 7000)
    ) {
  System.out.println(client.simpleRead("$OV_JOG"));
  System.out.println(client.simpleWrite("$OV_JOG", "90"));
}
```

Algorithm 4.3: Use-case for the new *simpleRead* and *simpleWrite* methods.

```
try (CrossComClient client = new CrossComClient("158.38.140.193", 7000)
    ) {
  double[] torques = client.readJointTorques();
}
```

Algorithm 4.4: Reading the actual torque for each axis, Java side.

program where the standard KRC kinematics is used to calculate the joint angles θ_d. Alternatively, a custom control algorithm can be implemented within the user program to calculate the joint values for the robot according to $\mathbf{x}_t$. Essentially, the custom control method has to implement classic inverse kinematic functions 2.9, 2.10.

Note that the possibility of implementing certain control features does not influence the design for the presented interface. Instead, *JOpenShowVar* extends the functionalities of the KRL language.

Additional functions. In order to simplify the low level communication protocol and improve reliability, some additional methods are provided with the *CrossComClient* class:

- the *simpleRead* and the *simpleWrite* methods are simpler versions of the *sendRequest* function. In particular, these methods do not execute any data parsing as opposed to the *sendRequest* method. They allow for an easier and faster access, as shown in the Algorithm 4.3 sketch box. The two new methods return a raw string without parsing the information. The aim of these two new methods is to provide an easy way to monitor the status of the robot makeing it possible to print the raw information returned from the KRC;
- the *readJointAngles* method uses the *readVariable* method to recursively retrieve the actual joint angles vector, θ_a, of the controlled robot, all at once;
- the *readJointTorques* method allows for monitoring the load of each joint actuator by retrieving the current torque of each axis of the arm, all at once. In particular, *readJointTorques* retrieves the global KRL array variable $TORQUE_AXIS_ACT and iteratively returns the current torque of each axis. This feedback signal is very important in order to improve manipulator dexterity and to achieve crucial functions like sensitive collision detection and compliant control actions. In the Algorithm 4.4 sketch box, a possible use-case is shown.

```
KRLAxis qd = new KRLAxis("MYAXIS");
//MYAXIS is defined manually in $CONFIG.DAT
qd.setA1ToA6(80, 10, -10, 20, 35, 32);
try (CrossComClient client = new CrossComClient("158.38.140.193", 7000)
    ) {
  client.writeVariable(qd);
}
```

Algorithm 4.5: Writing the robot's joint angles, Java side.

```
KRLPos pd = new KRLPos("MYPOS");
//MYPOS is defined manually in $CONFIG.DAT
pd.setXToZ(100, 12, 30);
pd.setAToC(-20, 25, 53);
try (CrossComClient client = new CrossComClient("158.38.140.193", 7000)
    ) {
  client.writeVariable(pd);
}
```

Algorithm 4.6: Writing the robot's end-effector position and orientation, Java side.

Some other possible high-level applications. In addition to these methods, some other high-level functions can be implemented by the user on top of the *JOpenShowVar* communication protocol. The following subsections provide the user with some guidelines that can be used to implement such methods. It should be noted that these possible high-level methods are not included in the *JOpenShowVar* library simply because their implementation depends on how the user declares the desired variables and the corresponding procedures on the KRC side.

Writing the robot's joint angles: a useful application that can be achieved consists of setting the joint angles, all at once. Let $q_d = [\theta_1, \theta_2, ..., \theta_{n_j}]^T$, with n_j being the number of joints, be the final desired joint configuration of the robot. q_d can be stored in a local *KRLAxis* variable. Remotely, on the KRC side, a corresponding *Axis* variable structure should be created with the same name in the predefined global system data list $CONFIG.DAT so that it can be accessed from the KRL program that will control the robot. By using the *writeVariable* method the entire desired configuration for the robot can be written at once as shown in the Algorithm 4.5 sketch box for a six DOFs robot.

Writing the robot's end-effector position and orientation: let $p_d = [x, y, z, \phi, \gamma, \psi]^T$ be the desired robot's end-effector position and orientation. p_d can be stored in a local *KRLPos* variable. Remotely, on the KRC side, a corresponding *Pos* variable structure should be created with the same name in the predefined global system data list $CONFIG.DAT so that it can be accessed from the KRL program that will control the robot. By using the *writeVariable* method the desired robot's end-effector position and orientation can be written at once as shown in the Algorithm 4.6 sketch box.

```
public void writeJointsPath(List<double[]> jointsPath)  throws
    IOException {
  try (CrossComClient client = new CrossComClient ("127.0.0.1", 7000))
      {
    int i = 1;
    for (double[] d : jointsPath) {
      client.sendRequest (new Request (0, "MYE6ARRAY [" + i++ + "]" "{
          A1" + d [0] + "A2" + d [1] + ", A3" + d [2] + "A4" + d [3] +
          "A5" + d [4] + "A6" + d [5] + "}"));
    }
  }
}
```

Algorithm 4.7: Generate a path (joint space), Java side.

```
DECL INT ROW    ;array index declaration
  FOR ROW = 1 TO 512
    PTP MYE6ARRAY[ROW]
  END FOR
```

Algorithm 4.8: Generate a path (joint space), KRC side.

Writing the robot's path (joint space): in a possible application scenario, it is often necessary to deal with paths defined in the joint space. Let $Q = [q_1, q_2, ..., q_n]^T$, where n is the number of desired joint configurations, which together make out the desired path in the joint space, and where $q_i \in \Re^{n_j}$. Since the new method *writeVariable* cannot handle arrays, each desired joint configuration has to be sent as a string with the *sendRequest* method. Basically, the *sendRequest* function is iteratively used to update a global array of E6AXIS on the KRC side. For each iteration, the new desired joint configurations are actuated with a PTP command. A possible use-case is suggested for a six DOFs robot. The Algorithm 4.7 sketch box shows a possible Java code, while The Algorithm 4.8 shows a possible implementation on the corresponding KRL side.

Writing the robot's path (Cartesian space): similarly, it can be useful to generate paths in the Cartesian space. Let $P = [p_1, p_2, ..., p_n]^T$, where n is the number of desired Cartesian configurations, which together make out the desired path in the Cartesian space and where $p_i \in \Re^6$. Also in this case, the *sendRequest* function is adopted. In particular, this time the *sendRequest* iteratively updates a global array of POS on the KRC side. For each iteration, the new desired Cartesian configurations are actuated with a PTP or LIN command. A possible use-case is suggested here. The Algorithm 4.9 sketch box shows the possible Java code, while The Algorithm 4.10 shows the possible implementation on the corresponding KRL program.

Setting binary outputs: in order to provide the possibility of setting binary outputs that can be used to open or close valves or control a gripper, another top level method could be added. It is not possible to set an output directly, but it can be done in a SPS loop cycle

```
public void writeCartesianPath(List<double[]> cartesianPath)  throws
    IOException {
  try (CrossComClient client = new CrossComClient ("127.0.0.1", 7000))
      {
    int i = 1;
    for (double[] d : cartesianPath) {
      client.sendRequest (new Request (0, "MYPOS [" + i++ + "]" "{X" +
          d [0] + "Y" + d [1] + ", Z" + "}"));
    }
  }
}
```

Algorithm 4.9: Generate a path (Cartesian space), Java side.

```
DECL INT ROW    ;array index declaration
  FOR ROW = 1 TO 512
    PTP MYPOS[ROW]
  END FOR
```

Algorithm 4.10: Generate a path (Cartesian space), KRC side.

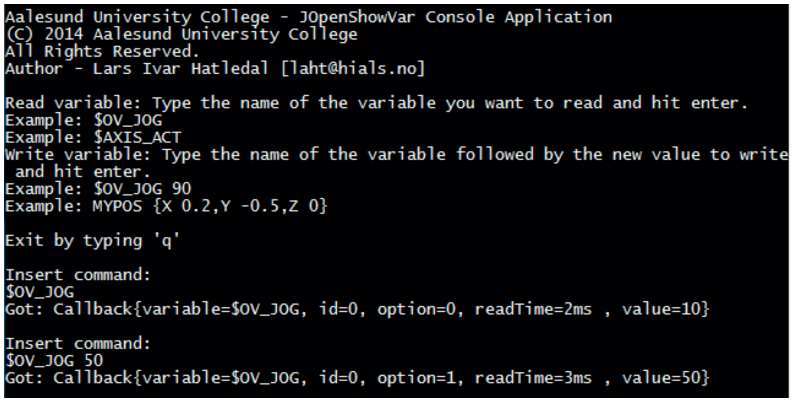
by means of global variables as shown in the Algorithm 4.11 sketch box:

Terminal and Graphical user interface. Another useful tool that comes with *JOpenShowVar* is a console terminal that provides read-write text-based access to the robot's data. It is particularly useful for system administration and debugging purposes. To read a variable, it is sufficient to type the name of the desired variable and press enter. From an implementation point of view, it uses the new *simpleRead* and *simpleWrite* methods. Figure 4.10-a shows a simple use-case.

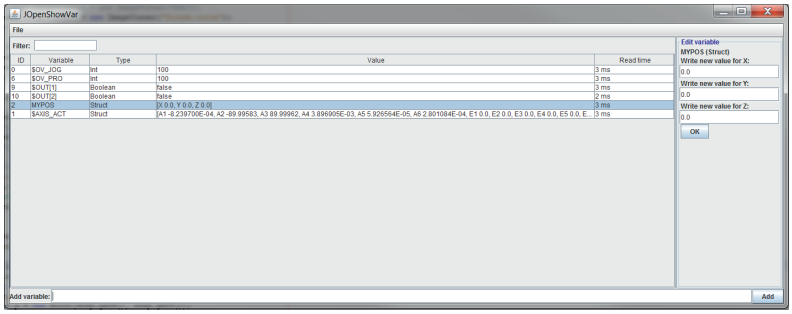
In addition, *JOpenShowVar* also offers a useful GUI that can be used to monitor the robot's state, visualise and manually set variables. A screen shot of this convenient tool is shown in Figure 4.10-b. This interface is very intuitive for the user.

```
SPS LOOP
...
$OUT[1] = OUTPUT_VARS[1]
...
$OUT[n] = OUTPUT_VARS[n]
...
SPS END LOOP
```

Algorithm 4.11: Setting binary outputs, KRC side.



(a)



(b)

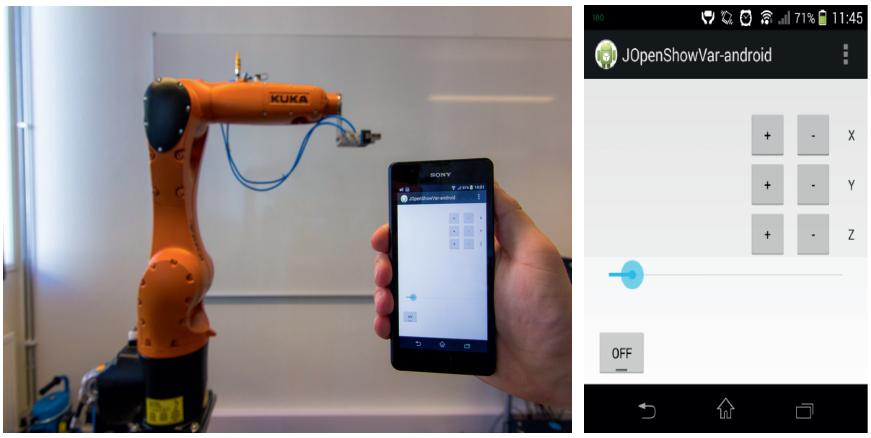
Figure 4.10: (a) *JOpenShowVar* terminal can be used for debugging purposes, (b) *JOpenShowVar* GUI can be used for monitoring the robot's state, visualise and manually set variables and structures.

Several case studies and related experiments and results concerning the proposed system integration are presented in the following.

4.2.2.2 Case Studies

Four case studies are presented to demonstrate the potential of *JOpenShowVar*. The first two case studies are open-loop applications, while the last two case studies describe the possibility of implementing closed-loop applications.

Case study 1: controlling the *Kuka KR 6 R900 SIXX* manipulator with an *Android* mobile device. To show the potential of the presented interface in controlling a *Kuka* industrial robot from an alternative input device, as a first case study, *JOpenShowVar* is used to control a *Kuka KR 6 R900 SIXX* manipulator with an *Android* mobile device.



(a)

Figure 4.11: Case study 1: (a) The *Kuka KR 6 R900 SIXX* manipulator, (b) the GUI of the Android mobile application used to control the arm.

In this case, an open-loop application is implemented by using the standard kinematics provided with the KRC. The *Kuka KR 6 R900 SIXX*, shown in Figure 4.11-a, is a 6 DOFs robotic arm with a slim design and a small footprint.

According to the operational scenario, an *Android* mobile application whose Graphic User Interface (GUI) is shown in Figure 4.11-b, is used to set the target position $\mathbf{x}_t$. By using the *writeVariable* method of *JOpenShowVar* this vector is forwarded to the *KUKAVARPROXY* and stored as a global value in a data structure. Finally, a KRL actuator program iteratively retrieves the new global data and uses the KRC kinematics to actuate the robot. The code of the KRL actuator program is shown in the Algorithm 4.12 sketch box.

For *Kuka* industrial robots, the idle time between motions can be shortened by executing the time-consuming arithmetic and logic instructions between motion commands while the robot is moving, i.e. processing them during the advance run (the instructions "run" in "advance" of the motion). Using the system variable $ADVANCE, it is possible to define the maximum number of motion blocks that the advance run can process ahead of the main run (the motion block currently being executed). Since the main loop of the Server program consists of only one instruction, the system variable $ADVANCE is initially set to 1 in order to avoid the unwanted execution of the same line of code. Inside the main loop, a relative movement is iteratively executed to the global variable MYPOS, which is the one that stores the target position. The key word C_PTP is used to approximate the movement. The approximate positioning instruction is executed in a time-optimised manner: there is always at least one axis moving with the programmed acceleration or velocity limits. The system simultaneously ensures that the permissible gear and motor torques for each axis are not exceeded. Furthermore, the higher motion profile, set by default, ensures motion that is optimised in terms of velocity and acceleration.

```
DEF ACTUATOR()
  INI
  PTP HOME Vel = 100 % DEFAULT
  $ADVANCE=1
  LOOP
    PTP_REL MYPOS C_PTP
  ENDLOOP
  PTP HOME Vel = 100 % DEFAULT
END
```

Algorithm 4.12: KRL actuator program for the case study 1.

Case study 2: a two-dimensional line-following task with the *Kuka KR 6 R900 SIXX* manipulator. In this case study, *JOpenShowVar* is adopted to perform a two-dimensional line-following task with the same *Kuka* robot used in the previous example. In this case, an open-loop off-line application is implemented by using the standard kinematics provided with the KRC. The considered task can be used for applications like advanced welding operations and similar. In this experiment, a camera is mounted on the robot's end-effector and can capture a photo of the desired line to be followed on a plane. This vision feedback is used for the off-line detection of the path before starting the movement. In particular, once a photo of the line to be followed is taken, the operator manually selects the desired initial and final points. Then, the *A* search algorithm* [104] is used to efficiently find a traversable path between these two points within the region covered by the desired line. The detected traversable path is sampled with a predefined resolution and the resulting samples are stored in an array variable. The same array is used as an input for the robot's end-effector to be actuated point by point. The experiment setup is shown in Figure 4.12.

Case study 3: controlling the *Kuka KR 6 R900 SIXX* manipulator with a *Leap Motion Controller*. In this case study, *JOpenShowVar* is used to control the same robot (from the previous case studies) in a closed-loop and with a custom control algorithm. This is done to highlight the potential of the presented interface in developing alternative control methods that do not use the standard kinematic model provided by *Kuka*. Moreover, a *Leap Motion Controller* [105], shown in Figure 4.13, is used as alternative input device to control the robot. The *Leap Motion Controller* is a small USB input device that supports hand and finger motions as input without requiring contact or touching. This controller is designed to be placed on a physical desktop, facing upwards. Using two monochromatic infra-red (IR) cameras and three IR light-emitting diodes (LEDs), the device observes a roughly hemispherical area, to a distance of about 1 meter. The LEDs generate a 3D pattern of dots of IR light and the cameras generate almost 300 frames per second of reflected data, which is then sent through a USB cable to the host computer, where it is analysed by the *Leap Motion Controller* software and can be retrieved using the *Leap Motion APIs*. While the *Leap Motion Controller* makes it possible to control all the joints

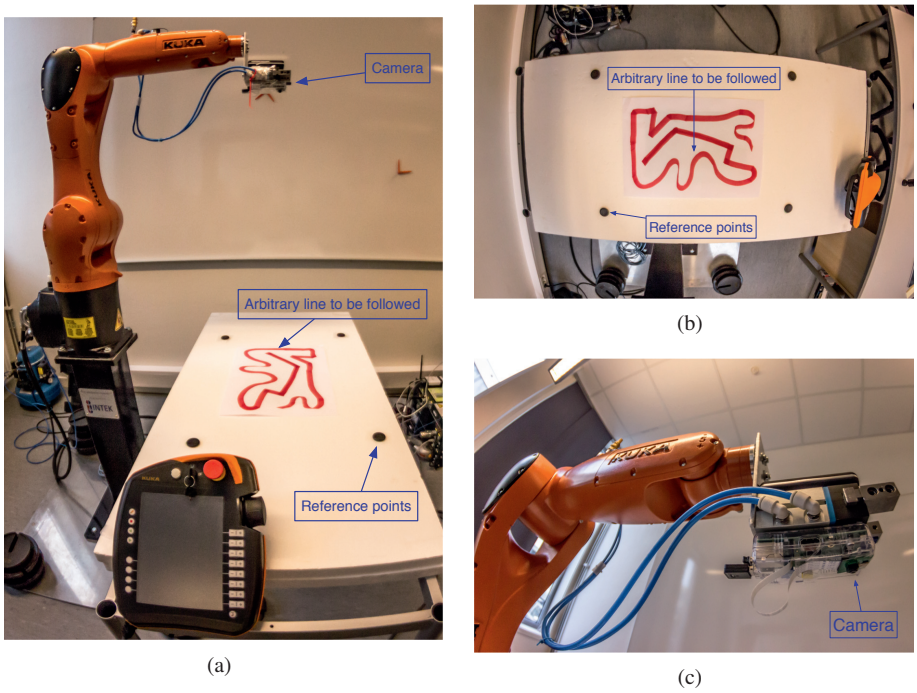


Figure 4.12: Case study 2: (a) The experiment setup for a two-dimensional line-following task with the *Kuka KR 6 R900 SIXX* manipulator, (b) the arbitrary line to be followed seen from the robot's view and (c) the camera mounted on the robot's end-effector.

of human hands, in this specific case study, only the DOFs of the wrist are used as an input signal to control the robot's end-effector. Each DOF of the wrist corresponds to a translational axis in the workspace of the robot to be controlled. When operating in position control mode, the input device works as a position proportional replica so that the wrist motion maps exactly to the motion of the robot's end-effector with constant speed, while, when operating in velocity control mode, a movement of the wrist in a particular direction will produce a translational motion in the same direction at a velocity proportional to the wrist displacement. In order for small vibrations not to affect the motion of the robot's end-effector, a small spherical imaginary volume with a diameter of about 8 cm is defined in the centre of the controller monitoring space. As long as the operator's wrist is located within this volume, the robot's end-effector does not move. The operator's hand has to be moved more than 4cm from the center of the monitoring space in order to generate a motion. Thanks to the modularity of the architecture, any other joystick or input device can be used without influencing the effectiveness of the proposed interface.

The user program runs on a remote computer and uses the *Leap Motion APIs* to retrieve

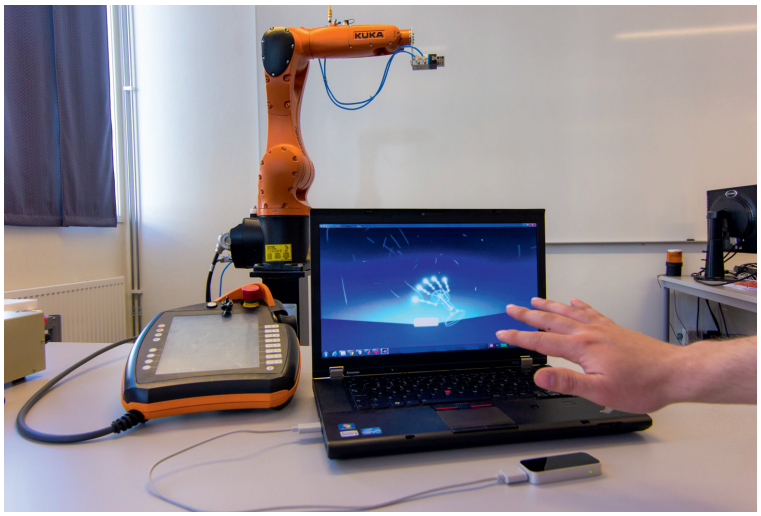


Figure 4.13: Case study 3: the *Leap Motion Controller* used to operate the *Kuka KR 6 R900 SIXX* manipulator.

the target position $\mathbf{x}_t$ according to the operational scenario. By using the *readVariable* method of *JOpenShowVar*, the actual joint angles θ_a are received. This data is used as input for the custom control algorithm. In this specific case study, the classical kinematic functions and the Jacobian method [106] are used to implement (2.9) and (2.10). Then, by using the *writeVariable* method of *JOpenShowVar* the target joint configuration θ_t is forwarded to the *KUKAVARPROXY* and stored as a global value in a structure. Finally, a KRL actuator program iteratively retrieves the new global data and actuates the robot.

The code of the KRL actuator program is shown in the Algorithm 4.13 sketch box. It should be noted that the variable MYAXIS is initialised to default values inside the INI fold. The system variable $ADVANCE is initially set to 1. Then the current joint values are assigned to a local structure variable named LOCAL. Inside the main loop, the desired joint angles are iteratively assigned to LOCAL, axis by axis. Finally, a PTP movement with C_PTP approximation is executed.

Case study 4: controlling the *Kuka KR 6 R900 SIXX* manipulator with an *omega.7* haptic device from *Force Dimension*. The aim of this fourth case study is to show the possibility of operating the robot and transferring the corresponding force feedback to the operator. For this purpose, a bidirectional coupling between a *Force Dimension omega.7* haptic device and the same *Kuka* robot used in the previous sections is established. In this case, an closed-loop application is implemented by using the standard kinematics provided with the KRC. The *omega.7* is a 7 DOF haptic interface with high precision active grasping capabilities and orientation sensing. Finely tuned to display perfect gravity compensation, the force-feedback gripper offers extraordinary haptic capabilities, enabling

```
DEF EXT_MOVE_AXIS()
  DECL AXIS LOCAL
  INI
  PTP HOME Vel = 100 % DEFAULT
  $ADVANCE=1
  LOCAL.A1 = $AXIS_ACT.A1
  ...
  LOCAL.A6 = $AXIS_ACT.A6
  LOOP
    LOCAL.A1 = LOCAL.A1 + MYAXIS.A1
    ...
    LOCAL.A6 = LOCAL.A6 + MYAXIS.A6
    PTP LOCAL C_PTP
  ENDLOOP
  PTP HOME Vel = 100 % DEFAULT
END
```

Algorithm 4.13: KRL actuator program for the case study 3.

instinctive interaction with complex haptic applications. In this case study, the principle of virtual works [106] is applied. According to this principle, the following equation is valid:

$$J^T \mathbf{F} = \tau, \tag{4.5}$$

where J is the Jacobian matrix of the arm, $\mathbf{F}$ is the vector of forces exerted from the robot's end-effector to the environment and τ is the vector of torques at the joints that can be retrieved by using the *readJointTorques* method. By applying this principle it is possible to simulate on the haptic device a force feedback proportional to the force that the robot's end-effector is supporting. The experiment setup is shown in Figure 4.14.

4.2.2.3 Experiment Results

Experiments related to the proposed case studies are carried out to test the proposed communication interface in terms of accuracy, performances and effectiveness.

Concerning the first and the third case studies, a demo video is available on-line at `https://youtu.be/6aZZAK4oyGg`.

The first case study highlights the potential of *JOpenShowVar* in remotely controlling a *Kuka* industrial robot from an *Android* mobile device. This possibility opens up to a variety of useful purposes including human interface applications and teleoperations. Nowadays, smartphones and tablets are becoming computationally more and more powerful. In this perspective, they are a perfect match with robots for developing alternative control systems. The use of smartphones and tablets in research and development is also found in other areas, as they represent a significant business opportunity for manufacturers, who need to consistently develop better hardware and operating systems. For these

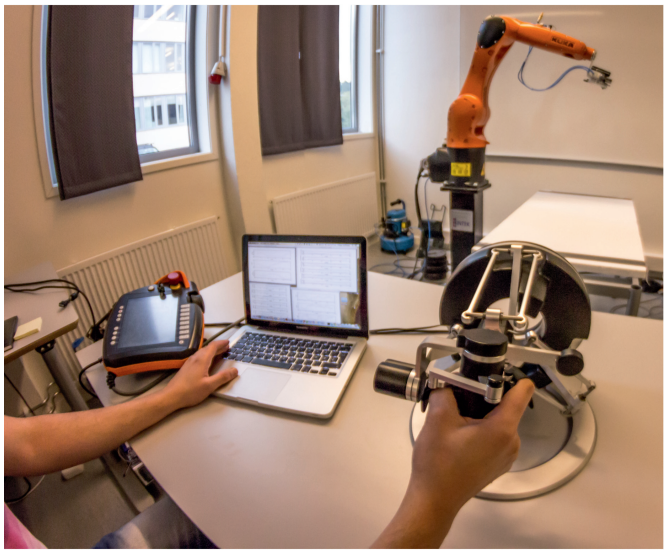


Figure 4.14: Case study 4: controlling the *Kuka KR 6 R900 SIXX* manipulator with a *omega.7* haptic device from *Force Dimension*.

reasons, these applications are very interesting and appealing in the forthcoming industrial applications.

Accuracy. Accuracy refers to the possibility of positioning the robot's end-effector at a desired target point within the workspace. Concerning the second case study, the line-following experiment is performed on a randomly generated line, drawn on a table. Figure 4.15 shows the detected line and the actual path followed by the robot's end-effector respectively. Once the line is detected, the robot executes the movement off-line in about $10s$ with a maximum position error less than 5 mm. This position error could be reduced even more by increasing the sampling resolution of the detected traversable path.

Performances. Within the particular case study of the *Leap Motion Controller* (case study 3), a real-time path tracking analysis of the Cartesian paths for X, Y and Z coordinates is performed, measuring the difference between the desired and actual position of the robot's end-effector. The results are shown in Figure 4.16.

Moreover, to assess the communication delay of *JOpenShowVar*, a time-delay analysis is carried out.

The considered time-delay represents the time for each message to be received, performed and notified to the client by the KRC. Particularly, this time-delay is obtained by considering the exact instant in which the request is dispatched from the client and the exact instant

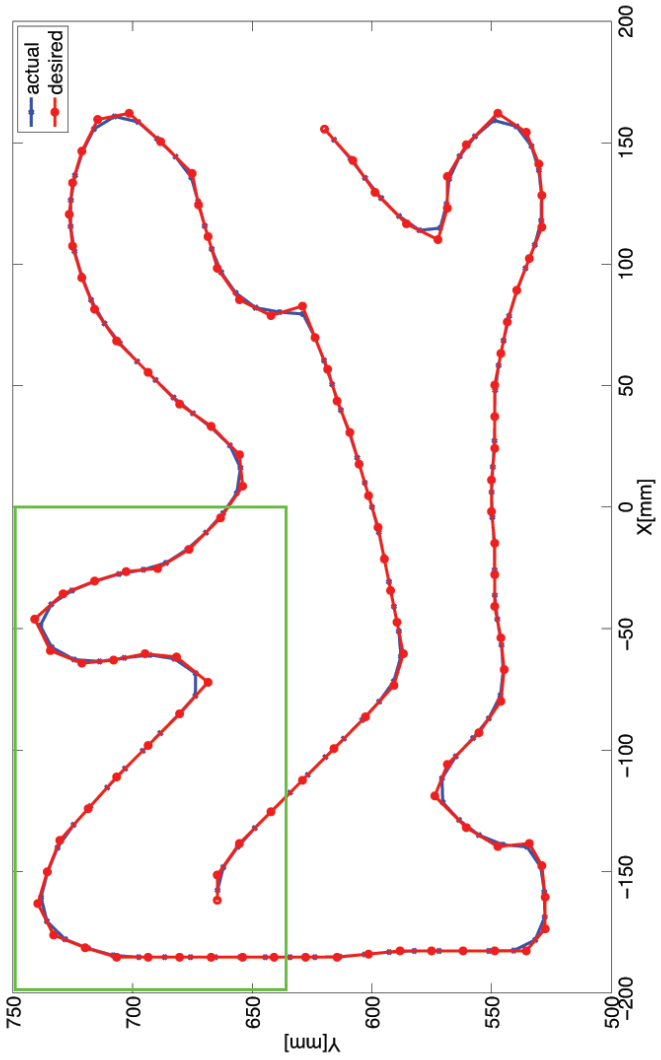
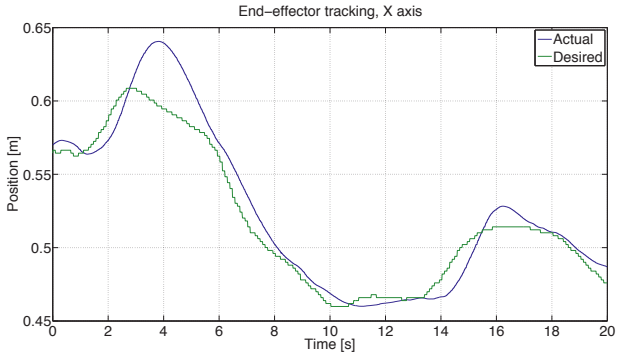
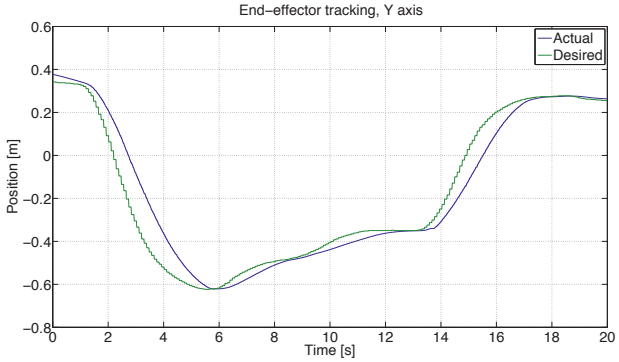


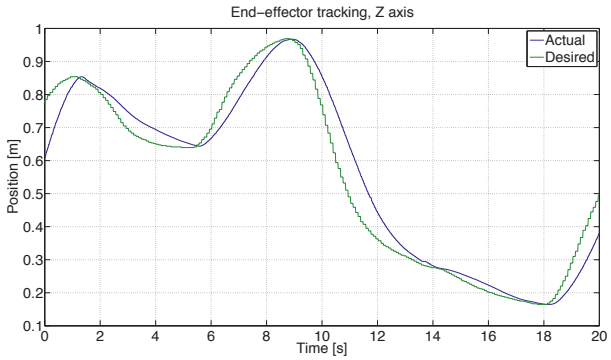
Figure 4.15: Case study 2: the detected line and the actual path followed by the robot's end-effector respectively.



(a)



(b)



(c)

Figure 4.16: Case study 3: path tracking for (a) the **X** coordinate, (b) the **Y** coordinate and (c) the **Z** coordinate.

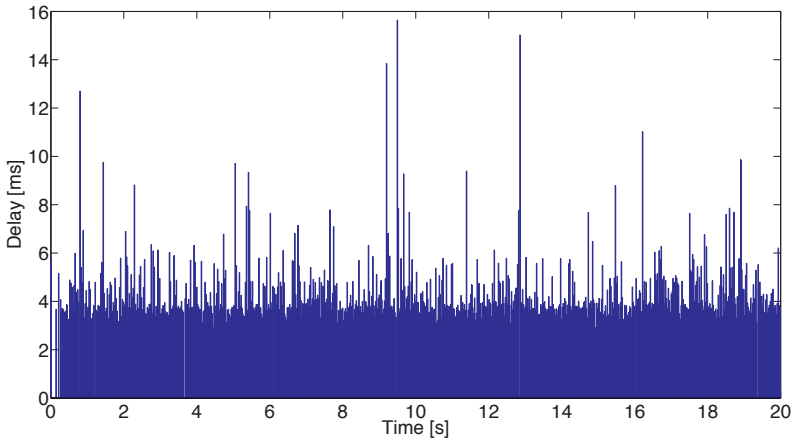


Figure 4.17: Case study 3: time-delay analysis for the corresponding Cartesian paths shown in Figure 4.16.

in which the information is received back from the client. It is not possible to exactly determine the time-delay mainly because *Kuka* has not released any information about it. During our experiments, a deterioration of the time-delay has usually been noticed when making the selection of a program and when the robot is engaged in some movements or there are several active interrupts. When considering the causes of the delay, it is possible to distinguish two main components that affect the access time for a variable to be either read or written from the client side:

- the time interval that is required for the *TCP/IP* protocol to transfer the information from the client to the server and then back to the client. This time component is non-deterministic;
- the time interval that is required for the *Kuka* controller to acquire the information from the robot. Also this time component is non-deterministic.

As already mentioned previously, the time-delay is not affected by the kind of access to be performed (whether it is a reading or a writing operation) or by the length of the message. Therefore, it is beneficial to aggregate several variables in logical structures when reading or writing data. By using data structures it is possible to simultaneously access several variables, thereby minimising the access time.

Considering the third case study, a time-delay analysis is carried out for the same Cartesian paths as shown in Figure 4.17. Even though there are a few spikes with a larger time interval, an average access time of 4.27 ms is obtained in this case. It should be noted that all the considered case studies are equally affected by similar communication delays except for the second case study which is performed off-line and therefore not presenting any run-time delays.

To further assess the performances of the proposed interface with regard to the communication delay, an additional experiment is performed. In particular, the possibility of developing alternative control methods is considered (as presented in case study 3). Any custom control algorithm that does not use the standard KRC kinematics must calculate the corresponding sampling point configurations for the desired end-effector's positions. In other words, each control algorithm works as a motion planner. In order to ensure smooth movements for the manipulators it is necessary to generate trajectories out of these given sampling points. A well-suited trajectory is the basic prerequisite for the design of a high-performance tracking controller and ensures that no kinematic nor dynamic limits are exceeded. Such a controller guarantees that the controlled robot will follow its specified path without drifting away. Therefore, feedback control has to be applied to be able to compensate external disturbances as well as disturbances from communication time delays. Note that time data is a free parameter because, as already mentioned, the sampling time of the mapping algorithm is not constant. A possible solution for generating well-suited trajectories consists of using a Proportional Integral Derivative (PID) controller for each joint. To tune the PID parameters, different methods can be used, such as the one proposed in [107]. The response of the PID controller that is adopted in the third case study is shown in Figure 4.18 for all the joints of the robot.

The interface provided by *JOpenShowVar* demonstrates a relatively fast reaction to the inputs and reasonable output error for research purposes, considering the dimension of the controlled model.

Effectiveness. Concerning the fourth case study, the aim is to show the possibility of operating the robot and transferring the corresponding force feedback to the operator. The plots in Figure 4.19-a and Figure 4.19-b show the actual position for the X, Y and Z axes as a result of the haptic input device's movements, operated by the user, and the corresponding joint angles, respectively. In this particular case, the operator manoeuvres the robot to lift the end effector up at first, then down again with a displacement also in the X and Y axes. In this case study, the input signal is not scaled to the robot's workspace since the haptic device is only used to set the direction of movement for the robot and to transfer the corresponding force feedback to the operator. Even though there is a delay between the input signal and the actual position, the results show that the system is quite responsive to the user's inputs. However, it should be noted that when the standard kinematics provided with the KRC is used (as for case study 4), the accuracy is worse than when using a custom control algorithm (as for case study 3). The reason for this performance degradation is mainly given by the internal implementation of the KRC. Figure 4.19-c and Figure 4.19-d show the torques applied to the robot's joints and the corresponding forces applied to the robot's end-effector, respectively. The operator also perceives a force feedback that is proportional to forces applied to the robot's end-effector.

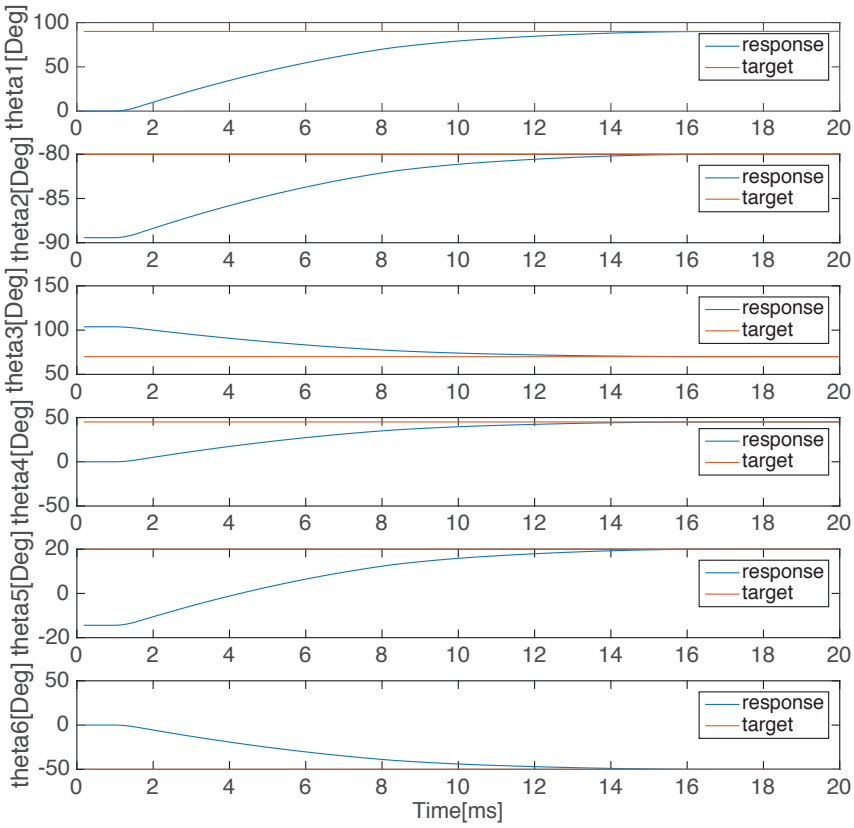


Figure 4.18: Case study 3: target and response of the adopted PID controller for all the joints of the robot.

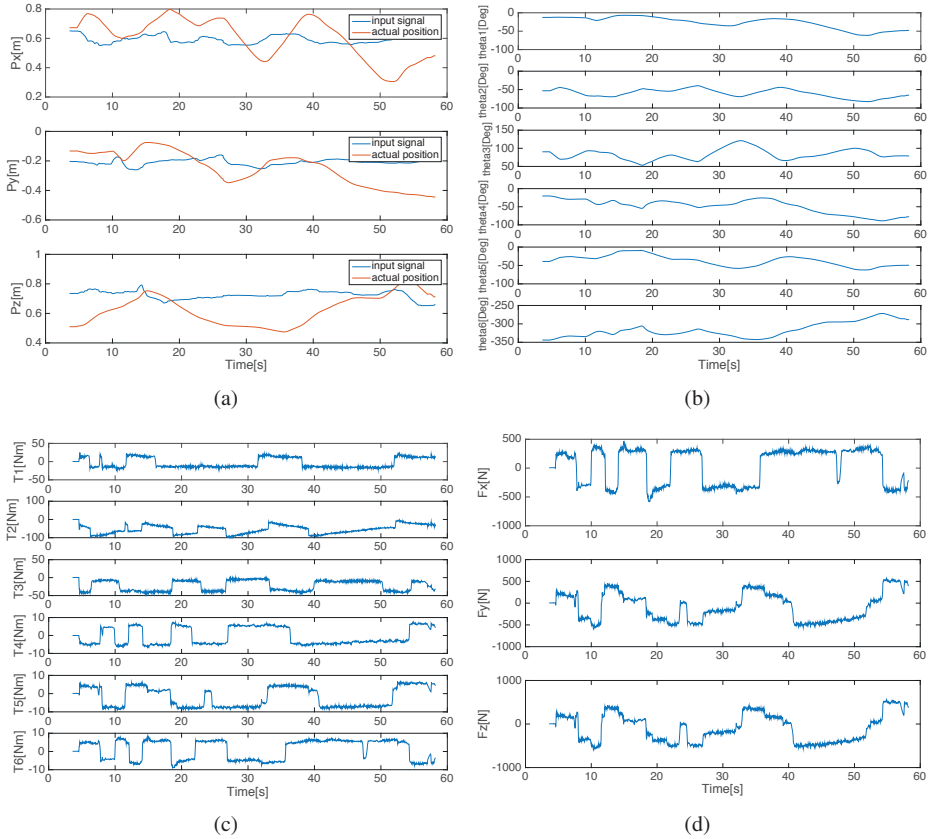


Figure 4.19: Case study 4: (a) actual position for the X, Y and Z axes as a result of the haptic input device's movements (note that, in this case study, the input signal is not scaled to the robot's workspace since the haptic device is only used to set the direction of movement for the robot and to transfer the corresponding force feedback to the operator), (b) the corresponding joint angles, (c) the corresponding joint torques and (d) the corresponding forces applied to the robot's end-effector.

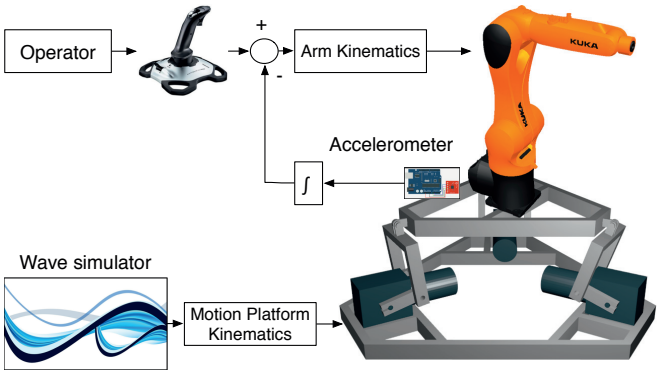


Figure 4.20: The proposed wave simulator and active heave compensation framework for demanding offshore crane operations.

4.2.3 A Integrated Wave Simulator and Active Heave Compensation Framework for Demanding Offshore Crane Operations

In the previous sections, the challenging problem of system integration has been addressed with regard to industrial robotic arms. Particularly, *JOpenShowVar*, a cross-platform communication interface that works with all *Kuka* robots, was presented. Taking full advantage of *JOpenShowVar* and in order to give researchers the possibility of testing alternative control algorithms for maritime cranes in a realistic and safe laboratory setup, a waves simulator and active heave compensation framework for demanding offshore crane operations is proposed in this section. The underlying idea is shown in Figure 4.20. The system is composed of an industrial robot, the *Kuka KR 6 R900 SIXX (KR AGILUS)* manipulator, and of a motion platform with three DOFs. This work focus on system integration. The motion platform allows the simulation of wave impacts, while the robotic arm can be manoeuvred by the user with a standard joystick or any other input device. An accelerometer is embedded on the platform in order to monitor the wave contribution. This same contribution is given as a negative input to the manipulator's control algorithm so that active heave compensation methods can be realised. It should be noted that only the heave compensation problem is addressed in this work, while all the issues related to rope pendulations are not considered (the robot on the platform is not equipped with any rope). A transparent user control interface can be implemented by using the proposed framework. In addition, the system can also be used for training purposes.

Regarding the system architecture, the presented framework is built on open-source software and hardware. Strict multi-threading criteria are applied to the control software in order to meet strict real-time requirements. The authors intend this work to be the first in a series of open-source designs to be released, and through the contributions of the open-source user community, result in a large number of design modifications and variations available to researchers. The official repository is available on-line at `https://`

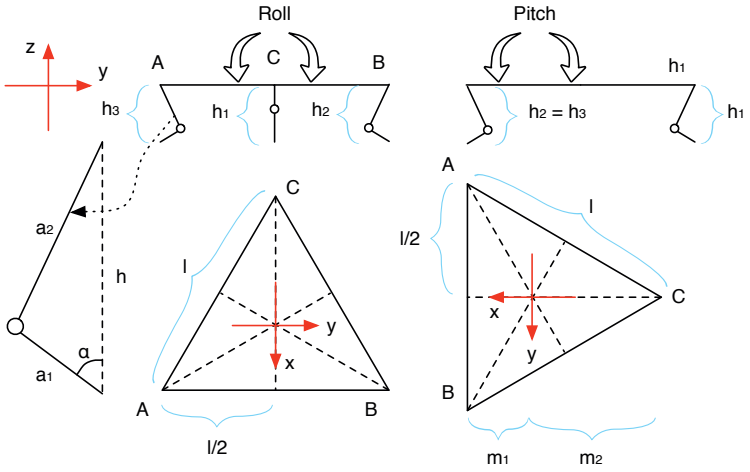


Figure 4.21: Geometric characteristics of the considered motion platform: $a_1 = 150$mm, $a_2 = 330$mm, $l = 1075$mm, $m_1 = 310.3$mm, $m_2 = 620.7$mm.

`github.com/aauc-mechlab/WaveSimulator`, along with several detailed class diagrams, all the mechanics, hardware schematics and demo videos.

4.2.3.1 System Architecture Integration

The main components of the proposed integrated system are presented. We first illustrate the considered motion platform from both a kinematic point of view as well as from a control point of view. Then the robotic arm is described focusing on the adopted control approach. Finally, the proposed integrated control system is depicted.

Motion platform. A 3D model of the adopted motion platform is available on our public repository. This model is a type of parallel robot that incorporates three DOFs. It consists of three arms connected to universal joints at the top base. Each joint is actuated by a motor allowing for controlling the corresponding corner of the top base. The rotation range of each joint is limited to 125° which corresponds to the joint pointing straight up, and the corresponding platform corner to have its maximum height. Any higher value of the joint angle would make the corresponding corner of the platform to decline again.

Referring to Figure 4.21, the design of the platform allows for movements along the Z axis (heave) and for rotations along the X and Y axes (roll and pitch, respectively). Given a desired heave position, h, each of the platform corners is raised or lowered to accommodate the position. For each corner of the equilateral triangle, h can be calculated as follows:

$$h = a_1 \cos(\alpha) + \sqrt{a_2^2 - a_1^2 \sin^2(\alpha)}, \tag{4.6}$$

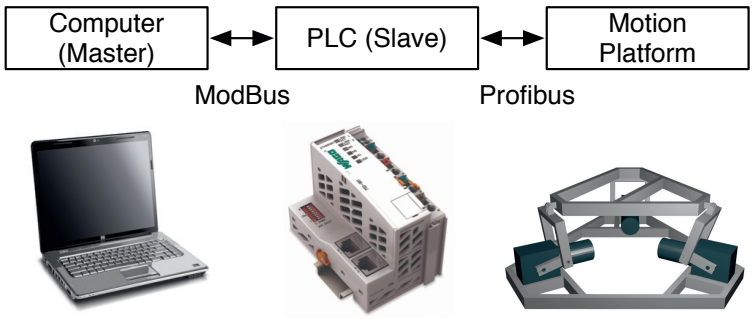


Figure 4.22: To control the motion platform, a master-slave architecture is used with the controller acting as a master and the PLC as a slave.

where a_1 is the lower arm, a_2 is the upper arm and α is the joint angle.

Concerning the roll movement, the height difference between h_2 and h_3 can be calculated as follows:

$$\Delta(h_2, h_3) = \sin(\phi)l, \tag{4.7}$$

where l is the length of the top base triangle and ϕ is the roll angle, which consequently can be found as:

$$\phi = \frac{\arcsin(\Delta(h_2, h_3))}{l}. \tag{4.8}$$

Concerning the pitch movements, the height of h_2 and h_3 can be calculated as follows:

$$h_2 = h_3 = -\sin(\theta)m_1, \tag{4.9}$$

where θ is the pitch angle and m_1 is shown in Figure 4.21. h_1 can be calculated as follows:

$$h_1 = \sin(\theta)m_2, \tag{4.10}$$

where m_2 is shown in Figure 4.21. Consequently, the pitch angle, θ, can be obtained as follows:

$$\theta = \arcsin(\frac{h_2, h_3 - h_1}{m_1 + m_2}). \tag{4.11}$$

In order to simulate a realistic application scenario, the control system that actuates the motion platform is independent from the control system that operates the robotic arm. In particular, the motion platform is controlled by using a hardware platform based on a commercial *Programmable Logic Controller* (PLC) [108]. The control architecture, which is shown in Figure 4.22, fully exploits the standard programming tools and the multi-tasking features offered by the PLC standard. By using the *Modbus* protocol [109], a master-slave pattern is set up with the controller acting as a master and the PLC as a slave. The three axes of the motion platform are driven by DC motors (203V). The motors are interfaced to a motor controller. In particular, a programmable power supply board

is used in order to avoid buying costly *H bridge* circuits. This board can be remotely controlled from the PLC via Profibus [110]. Besides, the motor revolution is controlled by means of inverters.

Robotic arm. The robotic arm that is placed on top of the presented motion platform is a *Kuka KR 6 R900 SIXX* manipulator. This manipulator is a 6 DOFs robotic arm with a slim design and a small footprint. The forward kinematics (FK) of this manipulator can be easily calculated by applying the standard Denavit-Hartenberg method [106]. In particular, the kinematics equations of a serial chain of 6 links like the considered robot, with joint parameters θ_i are given by:

$$T_A = {}^0_6T = \prod_{i=1}^{6} {}^{i-1}_{i}T(\theta_i), \tag{4.12}$$

where ${}^{i-1}_{i}T(\theta_i)$ is the general homogeneous transformation matrix from the frame of link i to link $i-1$.

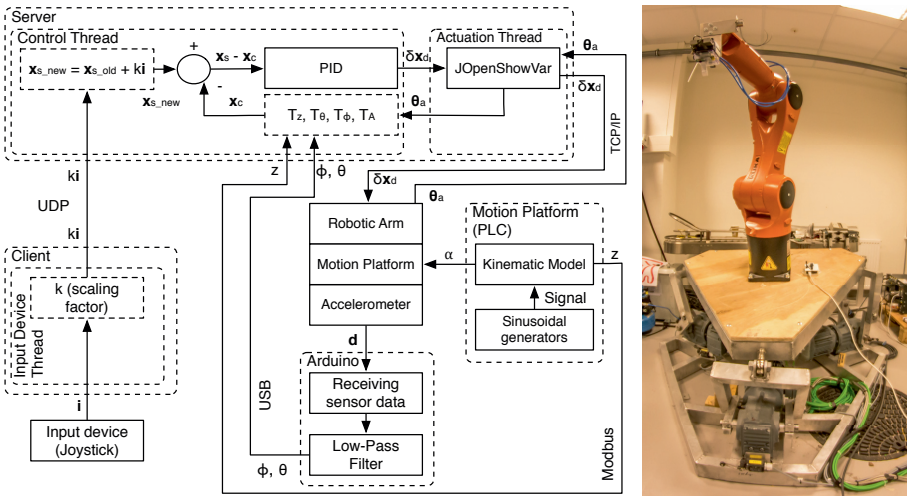
The robot can be operated by the user by means of a standard joystick. In order to efficiently control the robot, the open-source cross-platform communication interface provided by the previously introduced *JOpenShowVar* is used. This choice is motivated by the fact that *JOpenShowVar* allows researchers to implement alternative control algorithms according to current needs. In this study, the standard kinematics provided with the KRC is used to control the arm. The user program simply works as a driver for the input device and uses the *writeVariable* method of *JOpenShowVar* to forward the end-effector's target position, $\mathbf{x}_t$, to a *Kuka Robot Language* (KRL) program, where the standard KRC inverse kinematics is used to calculate the desired joint angles θ_d.

Integrated Control system. The integrated control system architecture is shown in Figure 4.23-a. It is a client-server architecture with the input device running as a client and communicating with a server where the logic of the control algorithm is implemented. The sever is implemented by following strict real-time criteria including multi-threading and synchronised methods. In the following, the key elements of the integrated control system will be presented referring to Figure 4.23-a.

Wave generation: random sinusoidal generators are used to reproduce the waves effect and to generate the input signal for the motion platform. The signal is generated as follows:

$$signal = \begin{bmatrix} A_h \sin(2\pi f t + \Omega) \\ A_\phi \sin(2\pi f t + \Omega) \\ A_\theta \sin(2\pi f t + \Omega) \end{bmatrix}, \tag{4.13}$$

where A_h is a random heave amplitude with uniform distribution in the range $[0, 150]$mm, A_ϕ is a random roll amplitude with uniform distribution in the range $[0, 100]$mm, A_θ is a random pitch amplitude with uniform distribution in the range $[0, 100]$mm, f is a random



(a) (b)

Figure 4.23: (a) the proposed integrated architecture: a client-server model is adopted. The server is implemented by following strict real-time criteria including multi-threading and synchronised methods. (b) the physical motion platform and the adopted robot

frequency variable with uniform distribution in the range $[0, 0.1]$Hz and Ω is a random phase variable with uniform distribution in the range $[-\pi, \pi]$. By using the kinematics of the platform, the corresponding joint angles, α, are calculated and used to actuate the motors.

Heave, roll and pitch detection: to monitor the platform roll and pitch movements, an accelerometer sensor is used. The raw data of the movements, $\mathbf{d}$, is collected and received by a controller board. In particular, an *Arduino Uno* board [77] based on the *ATmega328* micro-controller is used. *Arduino* is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. On the software side, *Arduino* provides a number of libraries to make programming the micro-controller easier. The choice of using *Arduino* boards makes the presented framework easy to maintain and makes it possible to add new features in the future. The raw data is filtered from noise and the roll and pitch angles, ϕ and θ, are sent to the server by using the Universal Serial Bus (USB). Concerning the heave movements, the displacement along the z axis is obtained directly by reading the actual angles of the motion platform and by applying the forward kinematics. Then z is sent to the server by using the *Modbus* protocol.

Input device: in this study, a standard joystick is used as a universal input device on the client side. Each degree of freedom of the joystick corresponds to a translational axis in the workspace of the manipulator to be controlled. The joystick works as a position proportional replica whose motion maps exactly to the motion of the arm. A movement of the joystick in a particular direction will produce a translational motion of the robot's

end-effector in the same direction, at a velocity proportional to the joystick displacement. When the operator's hand is removed from the joystick, the latter automatically returns to its starting point while the robot's end-effector keeps the last position. The joystick signal, $\mathbf{i}$, is scaled with a scaling factor, k, to fit the robot's workspace and then it is sent to the server by using the UDP protocol.

Control Server: in the following, the threads that run on the server side are described.

The *Control Thread* receives the following parameters:

- the scaled input signal from the joystick, $k\mathbf{i}$;
- the displacement, z, from the motion platform;
- the roll and pitch angles, ϕ and θ, from the accelerometer;
- the actual joint configuration from the manipulator, $\theta_{\mathbf{a}}$.

The current global robot's end-effector position, $\mathbf{x}_c$, can be obtained by using the following transformation matrix, T_c:

$$T_c = T_z T_\theta T_\phi T_A, \tag{4.14}$$

where T_z is the heave transformation matrix, T_θ is the pitch transformation matrix, T_ϕ is the roll transformation matrix and T_A is the arm transformation matrix.

At each control iteration, a set point, $\mathbf{x}_s$, is determined for the robot's end-effector as follows:

$$\mathbf{x}_{snew} = \mathbf{x}_{sold} + k\mathbf{i}, \tag{4.15}$$

where $\mathbf{x}_{snew}$ is the new set point and $\mathbf{x}_{sold}$ is the set point from the previous control iteration. The initial set point can be decided by the operator.

Successively, the difference between $\mathbf{x}_{snew}$ and $\mathbf{x}_c$ is calculated so that the corresponding sampling point configurations, $\delta\mathbf{x}_d$ (PID output), are obtained. In order to ensure smooth movements for the manipulators it is necessary to generate trajectories out of these given sampling points. A well-suited trajectory is the basic prerequisite for the design of a high-performance tracking controller and ensures that no kinematic nor dynamic limits are exceeded. Such a controller guarantees that the controlled robot will follow its specified path without drifting away. Therefore, feedback control has to be applied to be able to compensate for external disturbances as well as for disturbances from communication time delays. Note that time data is a free parameter because the sampling time of the mapping algorithm is generally not constant. As a possible solution for generating well-suited trajectories a Proportional Integral Derivative (PID) controller is used for each translational axis. To tune the PID parameters, different methods can be used, such as the one proposed in [107].

The *actuation thread* is used to communicate with the *Kuka* robot. This thread receives $\delta\mathbf{x}_d$ and uses the *writeVariable* method of *JOpenShowVar* to send the actuation values to the robot. In addition, the actual joint configuration, θ_a, is read by using the *writeVariable* method of *JOpenShowVar* and sent back to the *Control thread*.

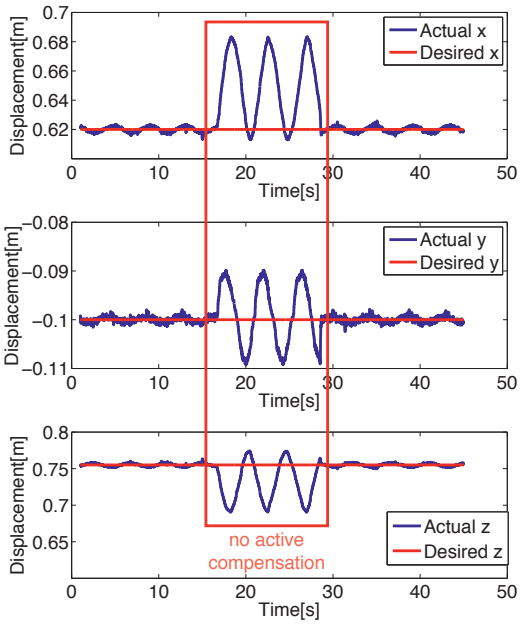


Figure 4.24: A time plot for the robot's end-effector position is performed.

This system integration give researchers the possibility of testing alternative control algorithms for maritime cranes in a realistic and safe laboratory setup. Related experiments and results are presented in the following.

4.2.3.2 Experiment Results

The physical motion platform and the adopted robot are shown in Figure 4.23-b. Related simulation are carried out in order to test the proposed framework. In detail, a time plot for the robot's end-effector position is performed. Figure 4.24 shows a time plot for the robot's end-effector position. Active compensation is performed except for the highlighted time segment. It should be noted that the end-effector's movements are significantly affected by the motion of the platform when no active compensation is used. Contrary, wave effects are almost suppressed when adopting the proposed control method.

Chapter Summary

In this chapter, the challenging problem of system integration was addressed.

Regarding modular robotic hands, our outcomes are summarised in the following.

- The integration between virtual and real modular prototypes was considered in this chapter. In particular, different issues about the integration between virtual and real modular manipulators was investigated. In this regard, implementation details about the previously introduced rapid-prototyping framework of *ModGrasp* were presented. These details include wiring schematics, guidelines on how to monitor the load of each joint actuator, specifications concerning the communication protocol and the adopted software libraries. This system integration allows for establishing a real-time one-to-one correspondence between virtual and physical prototypes.

Similarly, system integration challenges concerning the field of maritime cranes and robotic arms were succesively investigated in this chapter. Our findings are listed in the following.

- We proposed the integration of our previously introduced flexible control architecture for maritime cranes and robots with a simulation environment specifically designed for offshore applications is presented. The considered simulation environment is the Crane Simulator from the Offshore Simulation Centre AS (OSC) [57]. This system integration establishes the base for the research of alternative control algorithms, which can be efficiently tested in a realistic maritime simulation environment.

- Next, the system integration and control of real industrial robotic arms was studied in this chapter. Restricting the focus to *Kuka* industrial robots, we presented *JOpenShowVar*, an open-source cross-platform communication interface to *Kuka* industrial robots that allows for reading and writing variables and data structures of the controlled manipulators. *JOpenShowVar* opens up to a variety of possible applications making it possible to use different input devices, sensors and to develop alternative control methods. Four case studies are presented to demonstrate the potential of *JOpenShowVar*. The first two case studies are open-loop applications, while the last two case studies describe the possibility of implementing closed-loop applications. In the first case study, the proposed interface is used to make it possible for an *Android* mobile device to control a *Kuka KR 6 R900 SIXX (KR AGILUS)* manipulator. In the second case study, the same *Kuka* robot is used to perform a two-dimensional line-following task that can be used for applications like advanced welding operations and similar. In the third case study, a closed-loop application is developed to control the same manipulator with a *Leap Motion Controller* that supports hand and finger motions as input without requiring contact or touching. In the fourth case study, a bidirectional closed-loop coupling is established between a *Force Dimension omega.7* haptic device and the same *Kuka* manipulator. Related simulations are carried out to validate the efficiency and flexibility of the proposed communication interface.

- Finally, taking advantage of the possibilities offered by *JOpenShowVar*, the integration of our cranes and robots control system with a physical motion platform was developed. In particular, the system is composed of an industrial robot, the *Kuka KR 6 R900 SIXX (KR AGILUS)* manipulator, and of a motion platform with three DOFs. The motion platform allows the simulation of wave impacts, while the

robotic arm can be manoeuvred by the user with a standard joystick or any other input device. An accelerometer is embedded on the platform in order to monitor the wave contribution. This same contribution is given as a negative input to the manipulator's control algorithm so that active heave compensation methods can be realised. The proposed system integration gives researchers the possibility of testing alternative control algorithms for maritime cranes and robotic arms in a realistic and safe laboratory setup. Related simulation were carried out in order to test the proposed system.

CHAPTER 5

Benchmarking Different Control Methods

In this chapter, the challenging problem of assessing the performance of different control methods is addressed.

With regard to modular robotic hands, the computation of grasp quality indices is considered. Particular focus is put on the grasp measuring index adopted in our design method for modular grasping hands, which was presented in Chapter 2.

Similarly, the challenge of benchmarking different control methods is successively investigated in the field of maritime cranes. In detail, a benchmark suite for advanced control methods of maritime cranes is presented. This suite is transparently integrated with our control architecture allowing for modelling different manipulator models, all the corresponding hydraulic systems, various vessels, and the surrounding environment for visualisation. Different control methods can be easily implemented and tested. A set of routine tests, different cost functions, and metrics are provided – taking into account several factors, including position accuracy, energy consumption, quality, and safety for both the cranes and the surrounding environment. Each proposed routine test is task-oriented, and it systematically reproduces realistic on-board operation scenarios. The concept of operation profiles is introduced, allowing for defining different standard transporting and lifting operations. By considering task-oriented routines, this benchmark suite allows for comparing different control methods independently from the specific crane model to be controlled. Related simulations are carried out by using the proposed benchmarking framework. In particular, the two control methods, which were previously introduced in Chapter 3 (based on GAs and on PSO, respectively), are considered for an extensive comparison.

Contributions of this chapter: with regard to modular robotic hands, we contribute to this chapter by investigating the computation of grasp quality indices that can be used to assess different grasping methods. Specifically, we describe the grasp measuring index adopted in the design method for modular grasping hands, which was presented in Chapter 2. This grasp measuring criteria was formerly introduced in [58]. We thereby provide the motivation of this choice.

Alike, we contribute to the issue of benchmarking different control methods in the field of maritime cranes. A benchmark suite for advanced control methods of maritime cranes

is presented as one of the main contribution of this chapter. This benchmark suite is important because it allows for comparing different control methods independently from the specific crane model to be controlled.

Organization of this chapter: Following the same line of the thesis, this chapter is divided in two main sections. In Section 5.1, a review of different criteria that can be used for assessing the performance of modular grasping hands is presented. Particular emphasis is given to the quality index adopted by the authors for the previously presented design method for modular grasping hands. Similarly, in Section 5.2, the challenge of qualitatively comparing different control methods for maritime cranes and robot is considered. In detail, a benchmark suite for advanced control methods of maritime cranes is presented.

Publications: The results of this chapter concerning modular robotic hands are based on the paper [61]. The findings related to maritime cranes and robotic arms are based on the paper [111].

5.1 Benchmarking in the Field of Modular Robotic Hands

When trying to benchmark several robotic systems or different control methods, a standard set of performance metrics needs to be defined. With regard to modular robotic hands, a review of the quality measures proposed in the grasp literature to quantify the grasp quality was presented in [112]. The authors classified the quality measures into two groups according to the main aspect evaluated by the measure: the location of the contact points on the object or the hand configuration. The same authors also presented a review of the approaches that combine different quality measures from the two previous groups to obtain a global quality measure.

For the implementation of our design method for modular grasping hands, which was presented in Chapter 2, any quality criteria can be implemented without varying the structure of the proposed algorithm. However, most of the known quality measures do not consider any limit on the magnitude of the forces applied by the fingers. Thus, even when the obtained force-closure grasps can resist external perturbation wrenches with any direction, nothing is said about the magnitude of the perturbation that can be resisted. This means that in some cases the fingers may have to apply extremely large forces to resist small perturbations.

To overcome this issue, in our implementation, we adopted a quality criteria that is associated with the position of the contact points. In particular, we have used the index that was introduced by Ferrari and Canny in [58]. As a quality index, Ferrari and Canny considered the length of the radius of the largest inscribed sphere centered at the origin and entirely inside the *Grasp Wrench Space* (GWS). The GWS is the set of all wrenches that a grasp can resist if unit contact forces are applied at the contact points. It is given by

the convex hull of the elementary wrenches:

$$GWS = ConvexHull\left(\cup_{i=0}^{n}\{w_{i,1},\ldots,w_{i,k}\}\right), \tag{5.1}$$

where n is the number of contact points and k is the number of faces of the friction cone. The length of the radius of the largest inscribed sphere centered at the origin and contained within the *GWS* can be also seen as the magnitude of the largest worst-case disturbance wrench that a grasp is capable of resisting with a unit strength grip.

Several simulations have been carried out in order to test the design method proposed in Chapter 2. Efficient grasping configurations were found for grasping a set of daily objects. The adopted quality index was used to test the effectiveness of the obtained grasps. The corresponding simulations and results are presented in the following.

5.2 Benchmarking in the Field of Maritime Cranes

In field of robotic arms, at least a few benchmark suites and methods for estimating the efficiency of the considered control approach already exist. Contrarily, in the field of maritime cranes there is a lack of a universally recognised benchmarking method for assessing the system performance. This is the main reason why it currently is extremely difficult not only to compare results of different control approaches, but also to assess the quality of the research presented by the authors.

In this section, a methodology for performing simulation-based verification and benchmarking in the area of maritime cranes control is outlined. Different control methods can be transparently implemented and tested. The underlying idea consists of building a benchmark suite, as shown in Figure 5.1. The suite includes a set of routine tests, different cost functions, and metrics that take into account several factors including position accuracy, energy consumption, quality, and safety for both the cranes and the surrounding environment. A systematic approach is used to organise the presented metrics according to the adopted measuring method. Each proposed routine test is task-oriented and consistently reproduces realistic on-board operation scenarios. The concept of operation profiles is introduced, allowing for defining standard transporting and lifting operations. In the following, the key elements of the proposed benchmark suite are presented.

5.2.1 Benchmarks and Operational Profiles for Maritime Cranes

In this section, different quality measures and operational profiles are proposed. We classify the proposed quality measures into two groups, according to whether a direct or indirect method is used to assess the considered properties.

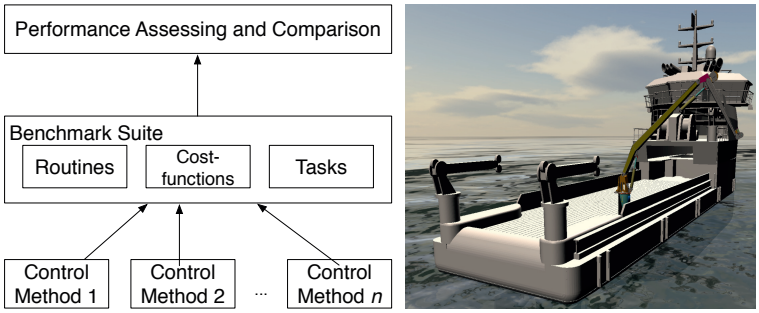


Figure 5.1: The realisation of a benchmark suite for advanced control methods of maritime cranes.

5.2.1.1 Direct Measures and Metrics

A direct measure of a property of interest for a process is a measure done on that process whose value alone indicates the extent of the property of interest.

Position and Joint Error. From a static point of view, the accuracy of a crane control system can be evaluated by using a composed cost function that assesses two different contributions: the end-effector position error, a, and the joint configuration error, b. The first contribution can be assessed by measuring the Euclidean distance between the target position, $\mathbf{x}_t$, and the calculated position, $\mathbf{x}_c$, from the adopted control method:

$$a = d(\mathbf{x}_t, \mathbf{x}_c) = |\mathbf{x}_t - \mathbf{x}_c|, \tag{5.2}$$

where $\mathbf{x}_c$ is calculated by using forward kinematics, while the target position is given by the desired position:

$$\mathbf{x}_t = \mathbf{x}_d. \tag{5.3}$$

The second component of the proposed cost function considers the change in joint angles between two consecutive solutions, and it can be calculated as the absolute difference between the target joints configuration, θ_t, and the calculated joint configuration, θ_c:

$$b = |\theta_t - \theta_c|, \tag{5.4}$$

where, the calculated joint configuration is the output from the adopted control method. This contribution is considered in order to avoid multiple solutions when considering redundant manipulator models. The complete cost function is calculated as:

$$\text{cost} = \alpha \text{a} + \beta \text{b}, \tag{5.5}$$

where $\alpha, \beta \in \Re^+$ are weighting factors, such that $\alpha + \beta = 1$.

Joint Torque. The joint torque effort, L, can be measured and used as a parameter of comparison between different crane control methods. For each particular control task, the joint effort can be assessed and calculated by using the following equation:

$$\begin{aligned} \mathbf{T} &= \int_0^{t_f} L(\theta_c, \dot{\theta}_c, \tau_c, t)\, dt \\ \text{subject to} \quad & \theta_c(0) = \theta_{c_0}, \dot{\theta}_c(0) = 0, \\ & \theta_c(t_f) = \theta_{c_f}, \dot{\theta}_c(t_f) = 0, \end{aligned} \tag{5.6}$$

where 0 is the initial time, t_f is the final time, θ_c is the calculated joint configuration, $\dot{\theta}_c$ is the calculated joint velocity, and τ_c is the calculated joint torque. In most cases, the effort, L, can be approximated by using the following equation:

$$L = \frac{1}{2}\|\tau_c\|^2. \tag{5.7}$$

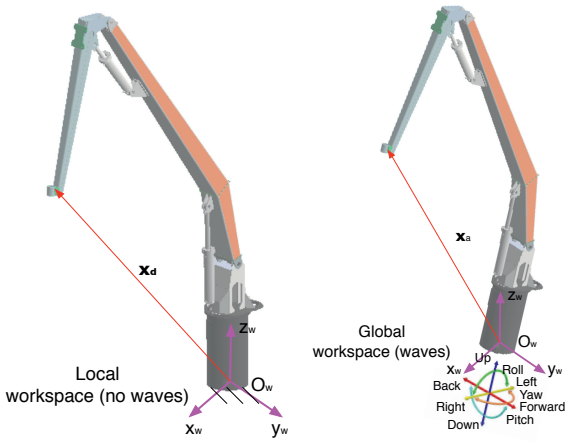
A crane control method that requires less torque for the joints when considering a particular control task, may be preferred to another control approach that uses a larger torque and consequently more energy to achieve the same task. In this regard, the joint torque can be also used as a cost function to be minimized in order to optimize planned motions when operating robotic arms [113].

5.2.1.2 Indirect Measures and Metrics

An indirect measure is a measure of a property of interest for a process that is performed by measuring either one or more different properties, either of that process or of other processes, and using those measures to determine the extent of the property of interest.

Under rough sea conditions, offshore activities involving crane operations result in many problems, including load sway, positioning accuracy, collision avoidance, and manipulation security. Unlike cranes mounted on fixed bases, offshore crane operations are significantly influenced by the ship motions resulting from currents and waves. The dynamic forces generated from the heave motion of the vessel and the sway movements of the pendulate load have extensive effects on the crane structure and the lifting wire. Concerning the heave effect, much research and many investigations have been done to help reduce the risks in offshore crane operations [114], [45], [44]. To monitor the ship movements, commercial offshore cranes usually adopt some motion detection units, e.g., Inertial Measurement Unit (IMU) and Motion Reference Unit (MRU). Then, based on this data input, a control system calculates how the actuators have to react to the movements. The actuators can be electric or hydraulic winch systems or hydraulic cylinders.

One possible indirect measure may consist in running a heave compensation method for the considered crane model to see which control method gives the best performances. This approach can be seen as a safety test. Any heave compensation method can be used to run this test. For the sake of clarity, we will present a possible approach.



(a) (b)

Figure 5.2: The idea behind the proposed heave compensation method: (a) the crane local workspace without waves contribution and (b) the global crane workspace with waves contribution.

Referring to Figure 5.2, two different cases can be distinguished. In Figure 5.2-a, the waves contribution is not considered. In order to consider the waves effect, a generalised model is depicted in Figure 5.2-b by simply adding 6 degrees of freedom (DOFs) to the base of the crane. When manoeuvring the crane, the operator sets the desired end-effector's position, $\mathbf{x}_d$, in the local workspace. However, because of the waves effect, it is necessary to consider the calculated end-effector's position, $\mathbf{x}_c$, in the global workspace. By considering the difference between $\mathbf{x}_d$ and $\mathbf{x}_c$, the position error can be calculated:

$$\delta\mathbf{x} = \mathbf{x}_d - \mathbf{x}_c. \tag{5.8}$$

This error can be used as the input for any desired crane control method. For instance, the classical Jacobian method [106] can be applied as follows:

$$\delta\theta = J^{-1}(\theta)\delta\mathbf{x}, \tag{5.9}$$

where $\delta\theta$ is the required joint velocity vector and $J(\theta)$ is the Jacobian matrix.

In order to indirectly evaluate the performance of the considered control method to compensate for the waves impact, one of the previously presented direct measures can be used.

5.2.1.3 Operational Profiles and Routine Tests

The reliability of a control system also depends on how the operator uses the crane. A good reliability estimation can be realised by testing the control methods to be compared

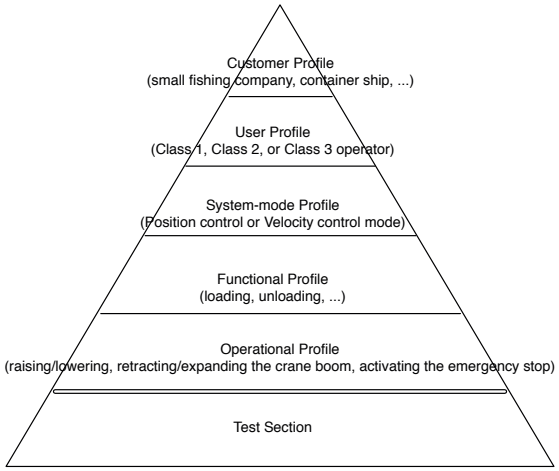


Figure 5.3: The process for developing an OP for a crane control system.

as if they were used in a real operation scenario. By borrowing the idea from the software engineering domain [115], we introduce the concept of *Operational Profile* (OP) as a quantitative characterisation of how the control methods are used by the operator. Different OPs can be created, and each of them can be repetitively executed by using the control methods that are to be compared. These tests can be run in a batch mode within a simulation environment.

A *Profile*, P, is a set of independent possibilities, called *states*, and their associated probability of occurrence. Each state defines a particular configuration for the crane – either in the joint or Cartesian space. For example, if state A occurs 60 percent of the time, B occurs 30 percent, and C occurs 10 percent, the profile is $P = [(A, 0.6), (B, 0.3), (C, 0.1)]$. An OP is the set of independent states that a control system performs and their associated probabilities.

The process for developing an OP for a crane control system is shown in Figure 5.3, and it involves one or more of the following steps.

Find the customer profile. As for any other product, a customer is the individual, group, or organisation that purchases the crane control system. A *customer profile* consists of an array of independent customer types. A *customer type* is one or more customers in a group that intend to use the control system in a relatively similar manner, and in a substantially different manner from other customer types. For instance, a *customer type* for a particular crane control system may vary from a small fishing company, which can use a small crane with little payloads, to a big container ship company, which may use a big and strong crane model to lift huge payloads. The *customer profile* is the set of customer types and their associated probabilities of using the control system.

Establish the user profile. The user of a crane control system may be different from the customer. The user is the crane operator. Several kinds of users may use the same control system for a specific model of crane. For instance, users can be categorised according to their level of experience. According to the crane regulations API-RP-2D [116] and latest addition, ABS, ANSI Standards, ASME B30 Rules, and OSHA rules, the crane operators can be classified as follows:

- Class 1 Operator: no restrictions or limitations;
- Class 2 Operator: limited to making lifts under 50% of crane's lifting capacity at any radius, limited to loading and unloading boats in calm seas only, supervised with lift over 50% of crane's lifting capacity at any radius, supervised with personnel lifts;
- Class 3 Operator: limited to making lifts under 50% of crane's lifting capacity at any radius (with direct supervision only), limited to loading and unloading boats in calm seas only (with direct supervision only), cannot make personnel lifts under any conditions (with or without direct supervision).

The *user profile* is the set of user types and their associated probabilities of using the control system.

Define the system-mode profile. A system-mode is a way that a control system can operate. For instance, a crane control system may essentially operate in two different modes, which are position and velocity control. System-modes can be thought of as independent operational scenarios. Normally, a control system allows for switching among modes sequentially. The system-mode profile is represented by the list of system modes and their corresponding occurrence probabilities.

Determine the functional profile. After a good system-mode profile has been developed, the focus should turn to evaluation of each system mode for the functions performed during that mode, and then assigning probabilities to each of the functions. Functions are essentially tasks that the crane operator can perform with the control system. For example, loading or unloading boats can be seen as a function. In order to assign occurrence probabilities, the best data source consists of usage measurements taken on the field. These measurements may be obtained from system logs or data storage devices. Occurrence probabilities computed from the historical data should be updated to account for new control functions, users, or environments.

Determine the operational profile itself. A function may include several *operations*. Examples of such *operations* include raising or lowering the crane boom, retracting or expanding the crane boom, or activating the emergency stop. These operations are shown in Figure 5.4 from a rigging/signalman point of view. In turn, *operations* are made up of

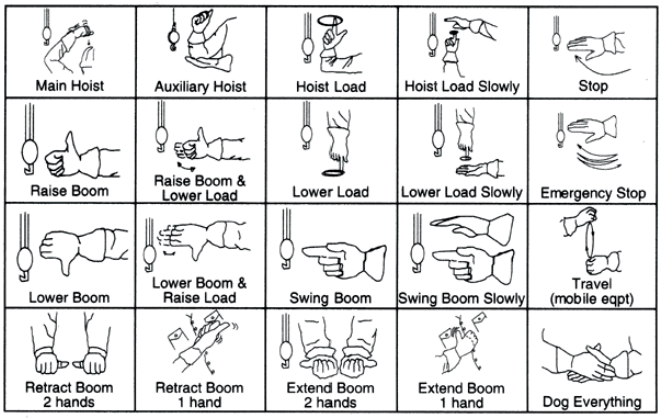


Figure 5.4: Different operations for a crane from a rigging/signalman point of view (International Crane Signals).

many *run categories*. In fact, the same *operation* may be performed with different specific requirements. For instance, the exact same *operation* of raising the crane boom may be required to be executed at low-speed or high-speed in terms of end-effector's velocity. Finally, each *run category* can be partitioned into different *run types* according to the particular input state. For instance, the exact same *operation* of raising the crane boom may start with different initial conditions in terms of payload and initial joint configuration.

Once the operational profile is determined, it can be used to run a routine test section against different crane control methods. It should be noted that the definition of *Operational Profiles* is very useful during the designing phase of a crane control system.

To show the potential of the proposed benchmark framework, the two alternative control approaches for maritime cranes and robots (one based on the use of GAs and the other one based on the use of PSO) presented in are considered to be extensively compared. The corresponding simulations and results are presented in the following.

5.2.2 Simulations and Results

In this section, related simulations are carried out by using the proposed benchmarking framework. In particular, the two control methods, which were previously introduced in Chapter 3 (based on GAs and on PSO, respectively), are considered for an extensive comparison. Each considered control method is used to control the same crane model. In detail, a knuckle boom crane is modelled including the hydraulic system for representing the mechanical properties. The actuators of the considered crane consist of one hydraulic motor at the foundation and base joint and two hydraulic cylinders positioned between the base and the boom, and the boom and the jib, respectively. A 500 kg payload is considered.

In order to consistently assess the performance of the two proposed methods, a common OP is defined so that the same test section can be run against the two methods to be compared. In particular, the *customer type* is a container ship company, which may use the crane to lift quite heavy payloads. The *user type* is a Class 1 Operator with no restrictions or limitations. The adopted *system-mode* is position control. The *function* of lifting a payload is considered. The following sequence of *operations* is taken into account: raising the crane boom, rotating the crane base, and lowering the crane boom. This sequence resembles the most common operations that are executed with the crane in order to handle and transfer objects from large container ships to smaller lighters or to the quays of the harbours. The input signal of this particular OP is generated by our framework by simply defining the corresponding path for the crane. During the generation of this input signal, the classical Jacobian method [106] is used to control the crane model and to generate the corresponding input samples. The input samples are stored as a temporal sequence. This sequence contains only input samples that produce reachable points in the crane's workspace.

5.2.2.1 Accuracy

The accuracy of the two proposed control methods is first analysed from a static point of view, by considering the position error, as seen in (5.2). The joint error is not considered since this specific crane model is not a redundant manipulator. For each of the methods to be compared, a trajectory tracking analysis of the Cartesian paths for X, Y, and Z coordinates is performed, and the results are shown in Figure 5.5 and in Figure 5.6 for the control method based on GA and the control method based on PSO, respectively. Each time plot shows the actual, the desired, and the calculated coordinates. It should be noted that the actual coordinates are obtained after the PID regulation process. This means that the dynamics of the crane are considered. These plots are qualitatively very similar, showing the effectiveness of the two considered methods. In addition, a time plot of the position error is shown for the two considered methods in Figure 5.7 and in Figure 5.8 for the control method based on GA and the control method based on PSO, respectively. The control method based on a PSO generally shows smaller position errors over time with smaller spikes.

5.2.2.2 Effectiveness

To assess the effectiveness of the two alternative control methods, the joint torque of each joint of the crane is monitored over time. The results are shown in Figure 5.9 and in Figure 5.10 for the control method based on GA and the control method based on PSO, respectively. It should be noted that the total execution time of the considered OP is similar for both the considered methods. Qualitatively, these time plots looks very similar.

Moreover, the total joint effort for the considered OP is calculated by using equation (5.6).

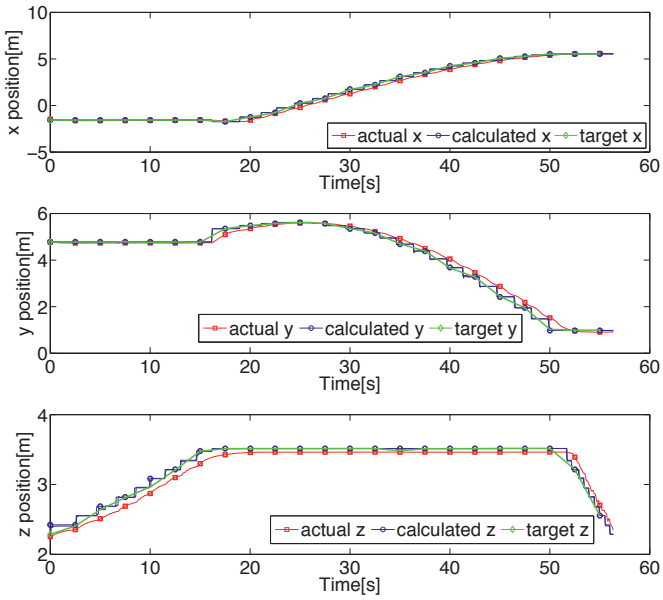


Figure 5.5: Trajectory tracking analysis of the Cartesian paths for X, Y, and Z coordinates while using the control method based on GA.

Table 5.1: Joint efforts calculated with Equation (5.6)

Control method	T_1[Nm]	T_2[Nm]	T_3[Nm]
Based on GA	0	7.9078E+12	0.0878E+12
Based on PSO	0	7.6934E+12	0.0773E+12

The results are shown in Table 5.1 for the two considered approaches. Practically, not such big differences between the two methods can be recognised. However, the control method based on a GA requires less torque for the joints over time for the considered OP. It is quite logical to suppose that bigger differences may be identified when comparing redundant manipulators.

5.2.2.3 Performances

To estimate the performance of the two considered alternative control methods, a heave compensation test is run. In particular, the same initial conditions are considered concerning the wave parameters, and the previously adopted OP is used as an input for the framework. A trajectory tracking analysis for the crane's end-effector position is performed for each considered control method showing the actual, the desired, and the calculated coordinates. Two cases are analysed: in the first case, no heave compensation is applied; in the second case, the presented heave compensation method is adopted. The results are

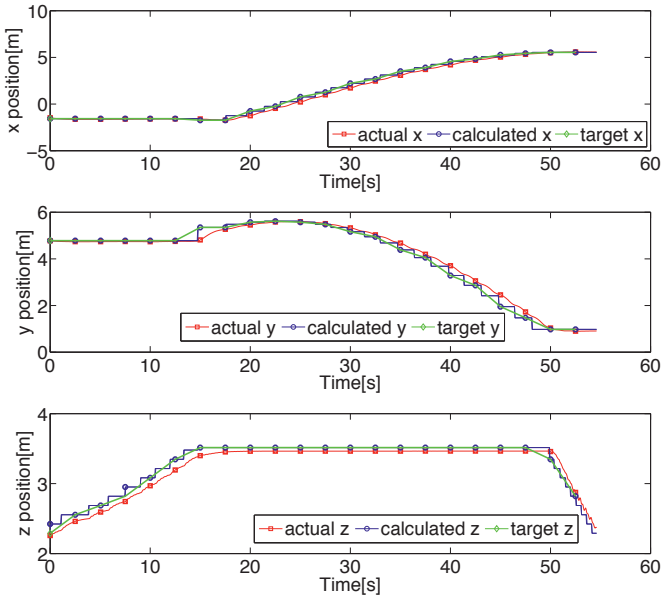


Figure 5.6: Trajectory tracking analysis of the Cartesian paths for X, Y, and Z coordinates while using the control method based on PSO.

shown in Figure 5.11 and in Figure 5.12 for the control method based on GA and for the control method based on PSO, respectively. Basically, the two considered methods share similar performances in terms of heave compensation. It should be noted that the calculated position of the crane's end-effector is efficiently compensated, while the actual position is closer to the target position when applying the heave compensation approach for both the considered methods.

Chapter Summary

In this chapter, we addressed the challenging problem of assessing and evaluating the performance of different control methods.

Regarding modular robotic hands, our results are summarised in the following.

- In this chapter, the computation of grasp quality indices was considered. Particularly, we described the grasp measuring index adopted in our design method for modular grasping hands, which was presented in Chapter 2. The adopted criteria allows for finding a limit on the magnitude of the forces applied by the fingers. Thus, the obtained force-closure grasps can not only resist external perturbation wrenches, but it is also possible to asses the magnitude of the perturbation that can

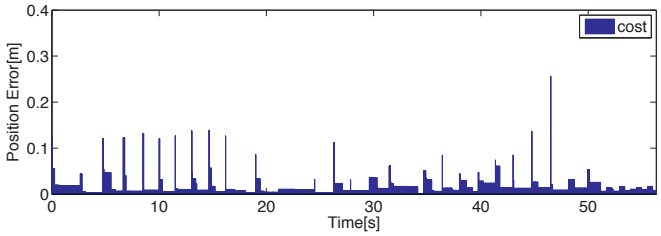


Figure 5.7: The time plot of the position error when using the control method based on GA.

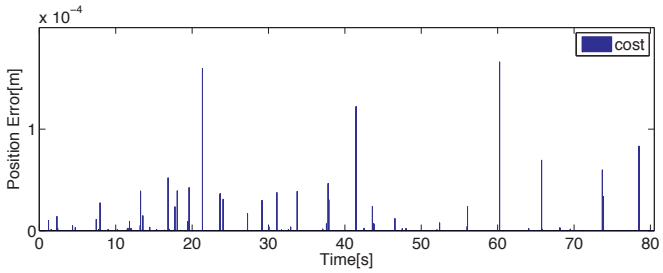


Figure 5.8: The time plot of the position error when using the control method based on PSO.

be resisted.

Similarly, the challenge of benchmarking different control methods was successively investigated in the field of maritime cranes. Our outcomes are listed in the following.

- We proposed a benchmark suite for advanced control methods of maritime cranes. This suite is transparently and consistently integrated with our control architecture allowing for modelling different manipulator models, all the corresponding hydraulic systems, various vessels, and the surrounding environment for visualisation. Different control methods can be easily implemented and tested. A set of routine tests, different cost functions, and metrics are provided. Several factor are taken into account including position accuracy, energy consumption, quality, and safety for both the cranes and the surrounding environment. Each proposed routine test is task-oriented, and it systematically reproduces realistic on-board operation scenarios. In order to make it possible to define different standard transporting and lifting operations, we introduced the concept of operation profiles. This benchmark suite allows for comparing different control methods independently from the specific crane model to be controlled. Related simulations were carried out. Specifically, the two control methods, which were previously introduced in Chapter 3 (based on GAs and on PSO, respectively), were considered for an extensive comparison.

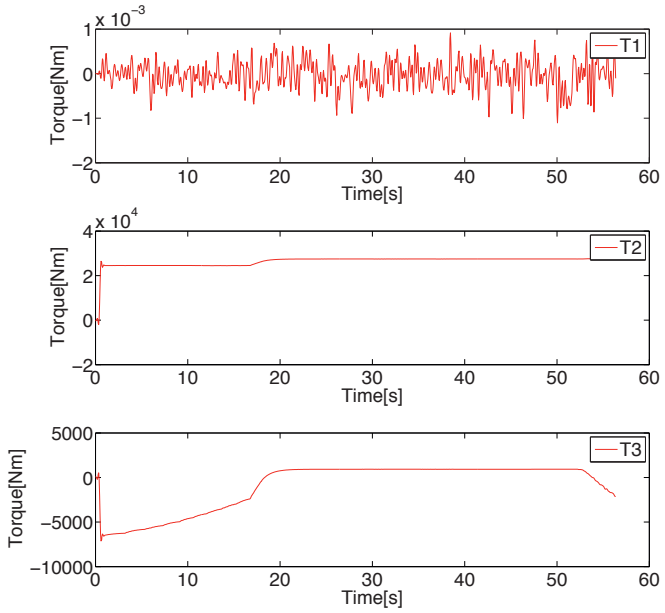


Figure 5.9: The torque time plot while using the control method based on GA.

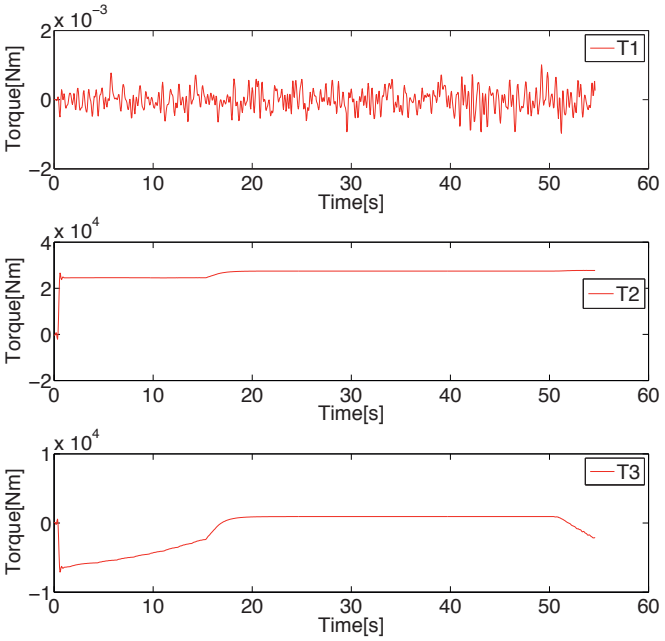
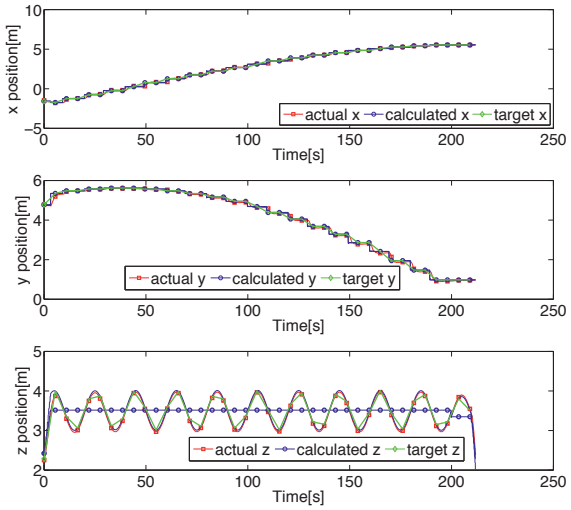
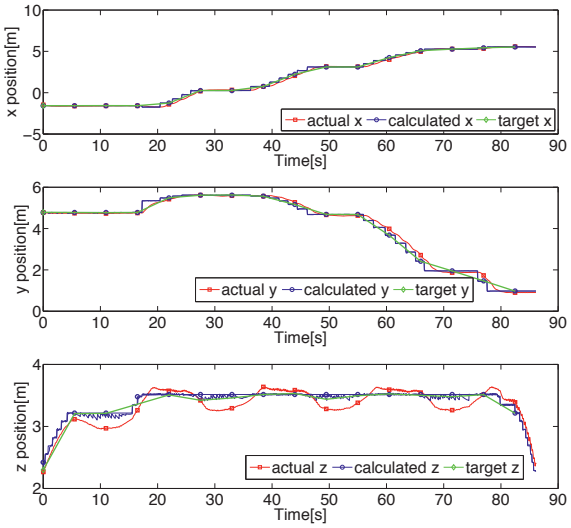


Figure 5.10: The torque time plot while using the control method based on PSO.

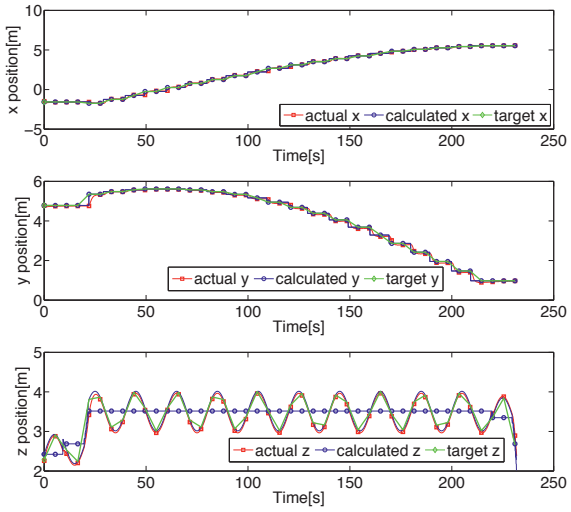


(a)

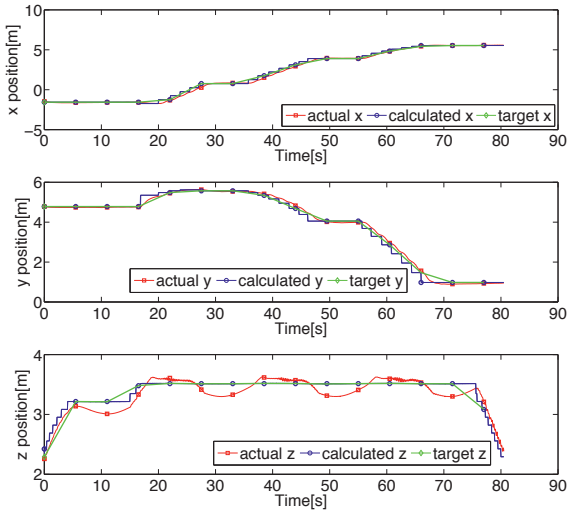


(b)

Figure 5.11: (a) The wave effect on the crane's end-effector without heave compensation and (b) with heave compensation when using the control method based on GA.



(a)



(b)

Figure 5.12: (a) The wave effect on the crane's end-effector without heave compensation and (b) with heave compensation when using the control method based on PSO.

CHAPTER 6

Conclusions and Future Challenges

Emphasising the similarities between robotic hands, maritime cranes and robotic arms, in this thesis several challenging common issues that currently affect the field of robotic manipulators were investigated.

Modular Robotic Hands. With regard to robotic hands, the problem of finding a trade-off between a simple gripper and more complex human like manipulators was initially investigated. In particular, the modular approach was adopted to obtain such flexibility. Modular robotic hands offer great advantages in terms of versatility since the obtained manipulators can be disassembled and reassembled to form new morphologies that are suitable for different tasks. Modularity also offers robustness to failures, considering that broken robot parts can be easily replaced. Another relevant feature offered by the modular approach is that the production cost can be considerably cut by building a specialised device capable of grasping objects by using only the number of actuators and DOFs required. However, in recent literature only few works investigated the possibility of developing an efficient design algorithm. To tackle this challenge, we initially investigated the possibility of designing a modular device that can adapt its structure to the object to grasp or to the task to fulfill. Particularly, we defined the guidelines for creating a modular hand capable of adapting its structure and functionality to the characteristics of an object or a set of objects to be grasped. These guidelines are based on the principle of *minimalism*, which consists in choosing the simplest mechanical structure, the minimum number of actuators, the simplest set of sensors and components that will do the desired job, or class of jobs. With this in mind, we introduced the concept of modular grasping to indicate when identical modules are used to build linkages in order to realize the grasping functions. From a mechanical point of view, even if it is not the most efficient grasping approach, the modular grasping still meets the requirements of standardization, modularization, extendibility and low cost. Based on this concept, a novel algorithm that determines effective modular configurations to get efficient grasps of given objects was presented. The goal of the proposed iterative procedure is to obtain a modular configuration that reaches a prefixed performance in terms of grasp quality using the least amount of modules possible. The resulting modular configurations are able to perform effective grasps that a human would consider "stable". Related simulations were carried

demonstrating the efficiency of our design algorithm.

Nonetheless, the proposed method is a simulation based approach and consequently it only allows for designing and testing virtual modular manipulators. To overcome this problem, *ModGrasp*, a combined virtual and physical design framework was successively presented. *ModGrasp* is an open-source virtual and physical rapid-prototyping framework that allows for the design, simulation and control of low-cost sensorised modular hands. The rapid-prototyping approach is combined with the modular concept making it possible to different model different manipulator configurations. Virtual and physical prototypes can transparently be linked in a real-time one-to-one correspondence. Different control algorithms can be implemented for the models. By using a low-cost sensing approach, functions for torque sensing at the joint level, sensitive collision detection and joint compliant control are possible. A 3-D visualization environment provides the user with an intuitive visual feedback. The main distinguished characteristic of this work with respect to the previous literature is that most of the previous works mainly focus on the mechanical construction process, while hardware, control and software prototyping are often neglected in the prototyping design. *ModGrasp* gives researchers the possibility of investigating different design and control methods by using an integrated mechanical, hardware and software rapid-prototyping environment.

Taking advantage of the possibilities offered by *ModGrasp*, the idea of developing alternative control methods for modular robotic hand was later considered by the authors. When considering a possible control method for simple modular robotic hands with a low number of DOFs, it may be sufficient to use conventional robotic control design tools and equations. However, when the complexity of the modular grasping model increases in terms of DOFs or when different modular configurations must be controlled independently of their specific morphology, a highly flexible and general control algorithm is needed. To address this challenge, a biologically-inspired control algorithm was considered. This control algorithm is based on the use of human synergies. To show the potential of this method, a three-fingered modular manipulator that can be controlled with brainwaves was presented as a case study. In particular, an EEG headset was used to monitor the user's levels of attention and meditation in order to generate a two-dimensional inputs to control the hand. Related simulations were carried out in order to test the considered synergistic control method within the particular case study.

While designing and developing *ModGrasp*, the authors focused on the challenge of system integration between virtual and real modular robotic hands. Particularly, we investigated different communication protocols and different hardware solutions. We implemented essential functions including the low-cost torque sensing feature, which makes it possible to realise crucial applications like sensitive collision detection and compliant control actions.

For the implementation of our design method, which allows to obtain effective modular hands to get efficient grasps, the challenge of benchmarking the effectiveness of different modular configurations was investigated. The generality of the proposed design method allows for using any quality criteria without varying the structure of the proposed algo-

rithm. However, most of the known quality measures do not consider any limit on the magnitude of the forces applied by the fingers. Thus, even when the obtained force-closure grasps can resist external perturbation wrenches with any direction, nothing is said about the magnitude of the perturbation that can be resisted. This means that in some cases the fingers may have to apply extremely large forces to resist small perturbations. For this reason, we decided to adopt a quality criteria that is associated with the position of the contact points.

Maritime Cranes. Regarding the study of maritime cranes, the possibility of overcoming the low control flexibility and non-standardisation problems that currently affect this field was considered. In particular, even though the operating environment can be very challenging, it is still quite common to use relatively simple control interfaces to perform offshore crane operations. Moreover, each input device can normally control only one specific crane model. When considering working efficiency and safety, this kind of control is extremely difficult to manage and extensive experience with high control skill levels is required of the operators. To address this challenges, we presented a general control architecture that allows for modelling, simulation and control of different models of maritime cranes and, more generally, robotic arms by using the same universal input device. Specifically, the proposed architecture makes it possible to transparently design flexible mapping procedures to map the fixed degrees of freedom of the universal input device to the variable degrees of freedom of the cranes or robots to be controlled. This process can be realised regardless of their differences in size, kinematic structure, body morphology, constraints, affordances and similar. Different mapping procedures can be designed and tested.

When designing a control algorithm for a crane or for a robotic arm, it is necessary to determine the kinematic properties of the system. One approach consists of studying the IK model of the manipulator to be controlled. This approach makes it possible to introduce analytical methods so that exact solutions for simple kinematic chains can be offered and solutions based on numerical methods can be proposed. However, inverse kinematics may lead to multiple solutions when considering arms that have redundant DOFs and singularity problems. It is also difficult to apply this approach when trying to control different manipulators with a universal input device due to the inherent complexity: each arm or crane to be controlled requires its own unique IK model. Using approaches that determine the kinematic properties by applying machine learning procedures or optimisation methods is a possible alternative solution to this problem. Taking advantage of the possibilities offered by our flexible control architecture, we presented two alternative control methods that allows for transparently control different manipulators and configurations. The first method is based on the use of GAs. The second method involves the use of PSO. Both methods derive the kinematic properties of the controlled arms by applying a machine learning procedure. In this way, our system automatically infers the mapping function for the different manipulators to be controlled. This approach only requires the FK model. Note that the unique feature of these methods compared to previous works is that the same set-up of the proposed algorithms is adopted independently of the manipulator being con-

trolled and whether the selected control mode is position or velocity. Moreover, when controlling each specific manipulator and once selecting the particular control mode, the same instance of the selected control method is continuously used; what differs are the semantics and the size of inputs and outputs which are dynamically and automatically set by the system.

Successively, we addressed some additional challenges considering the field of maritime cranes. Unlike cranes mounted on fixed bases, offshore crane operations are significantly influenced by the ship motions resulting from currents and waves. Due to the challenging crane operational scenario in real applications, several studies have been performed by using a computer-simulated environment. However, a simulation approach is always limited when compared to a realistic experimental setup. For this reason, we presented a wave simulator and active heave compensation framework for demanding offshore crane operations that makes it possible to reproduce the challenging operational scenario of controlling offshore cranes via a laboratory setup. The system is composed of an industrial robot and of a motion platform with three DOFs. The motion platform allows the simulation of wave impacts, while the robotic arm can be manoeuvred by the user with a standard joystick or any other input device. An accelerometer is embedded on the platform in order to monitor the wave contribution. This same contribution is given as a negative input to the manipulator's control algorithm so that active heave compensation methods can be realised. A transparent user control interface can be implemented by using the proposed framework. In addition, the system can also be used for training purposes.

Finally, the problem of assessing the performance of different control methods was also considered by the authors. Benchmarking as a means of objective comparison and competition among researchers is and has always been of great interest in robotics. As opposed to the field of robotic arms, where at least a few benchmark suites and methods for estimating the efficiency of the considered control approach already exist, in the field of maritime cranes there is a lack of a universally recognised benchmarking method for assessing the system performance. The reasons for this lack are different and include the fact that it is not possible to apply the same benchmarking methods that are common for other research fields to maritime cranes, mainly because of their size and complex operation scenarios. To tackle this challenge, the authors proposed a methodology for performing simulation-based verification and benchmarking in the area of maritime cranes control. In particular, a benchmark suite was developed. The suite includes a set of routine tests, different cost functions, and metrics that take into account several factors including position accuracy, energy consumption, quality, and safety for both the cranes and the surrounding environment. A systematic approach was used to organise the presented metrics according to the adopted measuring method. Each proposed routine test is task-oriented and consistently reproduces realistic on-board operation scenarios. The concept of operation profiles was introduced, allowing for defining standard transporting and lifting operations. The proposed benchmark suite was used to extensively test and compare the two alternative control methods presented by the authors and based on GAs and PSO, respectively.

Robotic Arms. With respect to robotic arms, the non-standardisation issue that currently affects this field was addressed. In particular, no common and standard control interface is present in industry and as such, manipulator control methods vary from one manipulator to another as necessity dictates. Not very many industrial manipulators have been relased with an open control interface, and of these few standard interfaces, most of them are limited to a small, specific set of models. Shifting the focus exclusively to *Kuka* industrial robots, the *Kuka Robot Language* (KRL) is the standard, proprietary language [28]. The language is text-based and offers data type declaration, specification of simple motions and interaction with tools and sensors by way of I/O operations. KRL programs can only be run on the *Kuka Robot Controller* (KRC), where programs are executed according to real-time constraints. The KRL interface, although quite easy to use in industrial applications, is also just as limited for research purposes. In particular, the KRL is tailored to the underlying controller and as such, only a fixed set of instructions is offered, which varies from controller to controller [29]. Advanced mathematical methods and calculations such as matrix operations, optimisation and filtering are not supported, which makes it considerably difficult to implement new control approaches. Third party libraries cannot be included directly from a KRL program, and as such, extending the code to include new instructions and functionalities is an arduous task. Furthermore, it is impossible to use external input devices directly. The standard workaround to the aformentioned problems is the inclusion of supplementary software packages provided by *Kuka*. Some examples of such packages are the *Kuka.RobotSensorInterface* [30], which allows sensor data to influence manipulator motion or program execution, and the *Kuka.Ethernet KRL XML* [30], a module that makes it possible to connect up to nine different external systems (such as sensors) to the robot controller. However, these supplementary software packages are accompanied by several significant drawbacks: I/O is limited, the set of functions present is small and purchasing these packages from *Kuka* is quite costly. To overcome these challenges, we presented *JOpenShowVar*, a Java open-source cross-platform communication interface that allows for reading and writing all the controlled manipulator variables. This interface allows researchers to use different input devices, sensors and to develop alternative control methods. *JOpenShowVar* library is compatible with all *Kuka* robots that use *KR C*4 or previous versions. Our framework works as a *middleware* between the user program and the KRL. Some high-level functions are also provided to enable angles and torques readings of the controlled manipulator. This feedback signal is very important in order to improve the manipulator dexterity and to achieve crucial functions like sensitive collision detection and compliant control actions with closed-loop control. *JOpenShowVar* is an open-source project.

Throughout this thesis, we have dealt with these challenges. Most of the developed systems are open-source projects. The authors believe that the key to maximising the long-term, macroeconomic benefits for the robotics industry and for academic robotics research relies on the closely integrated development of open content, open standards, and open source.

6.1 Summary of the Chapters

In this section, we present the conclusive remarks on each chapter of the thesis.

Summary of Chapter 1. In Chapter 1, we emphasised the similarities between robotic arms, maritime cranes and artificial hands. The concept of *robotic manipulator* was used as a general term to refer to all these devices. We stated the main challenges in order to provide an insight overview of the current State of the Art of the considered fields of study. In particular, several design challenges an control issues were highlighted. The low control flexibility and non- standardisation issues were also identified as crucial issues that currently effect robotic manipulators. Based on these challenging issues, we stated the main objectives of the thesis as:

- studying flexible design methods for robotic manipulators and developing a flexible control architecture to control different manipulator models;
- applying alternative and effective algorithms in order to control different manipulator models;
- realising the system integration between virtual and real environments and assessing the effectiveness of different control algorithms for robotic manipulators in a real application scenarios.

This objectives were followed throughout the remaining chapters of this thesis.

Summary of Chapter 2. In Chapter 2, we presented our initial results on developing efficient design methods and on the possibility of realising a flexible control architecture for robotic manipulators.

Regarding the study of modular robotic hands, we investigated the design of modular grasping grippers in order to find a trade-off between a simple gripper and more complex human like manipulators. Our design objective consisted in building a modular hand that can adapt its structure to the object to grasp or to the task to be fulfilled. In doing this, we chosen the simplest mechanical structure, the minimum number of actuators, the simplest set of sensors and components that will do the job, or class of jobs. Following these guidelines, a generalised modular model for robotic grasping was introduced. Based on this model, we successively presented a simulation-based algorithm that determines effective modular configurations to get efficient grasps of given objects. Related simulations were carried out in order to test the effectiveness of the proposed approach. To link the simulated models with real modular hands, we later developed *ModGrasp*, an open-source virtual and physical rapid-prototyping framework for designing low-cost sensorised modular grasping hands.

With regard to maritime cranes and robotic arms, the low control flexibility and non-standardisation issues were addressed. In particular, a generalised manipulator model

was initially considered. The generalised model consists of a kinematic chain that can be controlled by setting the position or the velocity of the joints. Based on this model, we presented a flexible control architecture that allows for modelling, simulation and control of different models of maritime cranes and, more generally, robotic arms by using the same universal input device regardless of their differences in size, kinematic structure, degrees of freedom, body morphology, constraints and affordances.

Summary of Chapter 3. In Chapter 3, we presented two alternative and effective algorithms that are able to scale and control different manipulators regardless of differences in size, kinematic structure, DOFs, body morphology, constraints and affordances.

When considering modular robotic hands, there is a challenge that arises despite the simplicity of the model. With the increase in the number of hand fingers and modules, the modular device becomes rival to the human hand in terms of complexity. To address this issue, a novel control method was presented based on a a biologically-inspired approach. This method is based on the use of human *synergies* and on the resulting dimensional reduction of the control problem. The adopted method was implemented based on *ModGrasp*, our virtual and physical rapid-prototyping framework for modular robotic hands which was previously presented in Chapter 2. As a case study, a mind-controlled, low-cost modular manipulator was presented along with related simulations.

In a similar way, we also investigated the possibility of developing alternative control methods concerning the field of maritime cranes and robotic arms. The need for alternative control methods is motivated by the fact that the classical approach of studying the IK model may lead to multiple solutions when considering arms with redundant DOFs. Additionally, singularity problems could arise. Furthermore, this method could hardily be applied when considering to control different manipulators using a universal input device because several IK models would be needed: one for each arm or crane to be controlled. To tackle this challenge, two alternative control methods were presented aiming to transparently control different manipulators and configurations. The first method is based on the use of GAs, while the second method involves the use of PSO. Both methods derives the kinematic properties of the controlled manipulators by applying an optimisation procedure. Related simulations were carried out.

Summary of Chapter 4. In Chapter 4, we tackled the challenging problem of system integration.

Considering modular robotic hands, we addressed the challenge of integrating virtual and real prototypes. In particular, we went through the implementation details concerning the integration of real modular manipulators with the previously introduced rapid-prototyping framework of *ModGrasp*. This system integration allows for establishing a real-time one-to-one correspondence between virtual and physical prototypes. A variety of possible application scenarios are possible.

Correspondingly, the system integration challenge was successively investigated in the field of maritime cranes and robotic arms. In particular, we presented the system integration of our flexible control architecture for maritime cranes and robots, which was introduced in Chapter 2, with a simulation environment specifically designed for offshore applications was presented. The considered simulation environment is the Crane Simulator from the Offshore Simulation Centre AS (OSC) [57]. This system integration represents a relevant contribution because it establishes the base for the research of alternative control algorithms, which can be efficiently tested in a realistic maritime simulation environment.

Next, we investigated the system integration and control of real industrial robotic arms, aiming to tackle the lack of standard controlling interfaces. In particular, we focused on the control of *Kuka* industrial robots. In this regard, *JOpenShowVar*, an open-source cross-platform communication interface to *Kuka* industrial robots was presented. *JOpenShowVar* allows for reading and writing variables and data structures of the controlled manipulators. The proposed interface opens up to a variety of possible applications making it possible to use different input devices, sensors and to develop alternative control methods. Different case studies were presented to demonstrate the potential of *JOpenShowVar* for both research and industrial applications.

Based on the new possibilities offered by *JOpenShowVar*, the integration of our cranes and robots control system with a physical motion platform was successively studied. In particular, a waves simulator and active heave compensation framework for demanding offshore crane operations was developed. The system consists of a motion platform, which is used to simulate the wave impacts, and a *Kuka* robotic arm, which can be manoeuvred by the user with a standard joystick or any other input device. An accelerometer is embedded on the platform in order to monitor the wave contribution. This same contribution is given as a negative input to the manipulator's control algorithm so that active heave compensation methods can be realised. This system integration give researchers the possibility of testing alternative control algorithms for maritime cranes in a realistic and safe laboratory setup.

Summary of Chapter 5. In Chapter 5, we explored the challenging issue of assessing the performance of different control methods.

Considering modular robotic hands, we considered the computation of grasp quality indices. Particularly, we analysed the grasp measuring index adopted in our design method for modular grasping hands, which was presented in Chapter 2. In our implementation, we adopted a quality criteria that is associated with the position of the contact points. In particular, we have used the index that was introduced by Ferrari and Canny in [58].

In a similar way, the challenge of benchmarking different control methods was successively investigated in the field of maritime cranes. This study is motivated by the fact that there is a lack of a universally recognised benchmarking method for assessing the system performance in this field. It currently is extremely difficult not only to compare results of

different control approaches, but also to assess the quality of the research presented by the authors. To address this challenge, we presented a benchmark suite for advanced control methods of maritime cranes. This suite is transparently integrated with our control architecture, which was presented in Chapter 2, allowing for modelling different manipulator models, all the corresponding hydraulic systems, various vessels, and the surrounding environment for visualisation. The system offers a set of routine tests, different cost functions, and metrics. Several factors are integrated, including position accuracy, energy consumption, quality, and safety for both the cranes and the surrounding environment. Each proposed routine test is task-oriented, and it systematically reproduces realistic onboard operation scenarios. The concept of operation profiles was included in the system, allowing for defining different standard transporting and lifting operations. By considering task-oriented routines, this benchmark suite allows for comparing different control methods independently from the specific crane model to be controlled. Related simulations were carried out by using the proposed benchmarking framework. In particular, the two control methods, which were previously introduced in Chapter 3 (based on GAs and on PSO, respectively), were considered for an extensive comparison.

6.2 Future Challenges

In this section, we investigate the future challenges concerning modular robotic hands, maritime cranes and robotic arms relative to the topics considered in this thesis. In particular, we point out some possible improvements and future work related to the main themes that were studied throughout this thesis.

Modular Robotic Hands. With regard to the flexible design method, which was proposed in Chapter 2, a consideration can be done. The proposed algorithm required a non trivial computational time. The most demanding part is the grasp planning phase. The amount of time required to complete this phase depends on the planner used, on the complexity of the object to grasp and on the number of modules involved. However, the structure of the proposed algorithm allows using different planners. Therefore, in the future more efficient planners may be implemented in order to reduce the execution time. In addition, task-oriented quality measures like those presented in [117] may be used or combined with traditional metrics (like the one described in Chapter 5) in order to further exploit the flexibility of the modular approach.

Considering, *ModGrasp*, the virtual and physical rapid-prototyping framework presented in Chapter 2, the authors intend this work to be an open platform for the open-source research community. In the future, new control methods may be implemented and compared with the one presented in Chapter 3. However, the simulation environment is still in the early stages of development and currently, only free-hand motions are possible. In the future, integration with a physics engine would allow for the simulation of controllable forces, object displacements, manipulability analysis and the addition of other grasp

quality measures so that the system integration presented in Chapter 4 can be improved even more.

Maritime Cranes and Robotic Arms. Concerning the control architecture for maritime cranes and robotic arms, which was presented in Chapter 2 and integrated with the Crane Simulator from the Offshore Simulation Centre AS (OSC) [57] in Chapter 4, the flexibility of the framework was proven. However, in this preliminary work, the nature of the manipulator actuators – whether they are hydraulic, pneumatic, electric, or mechanical – was not considered. This is a quite relevant aspect. It is important to include the nature of the manipulator actuators in the models to be controlled. In addition, when modelling maritime cranes, it is also relevant to consider the vessel models and the surrounding environment. In this respect, the system to be designed can be seen as a complex cyber-physical system (CPS) [118]. In order to realise such a CPS, it is necessary to combine different heterogeneous models. Unlike more traditional systems, a full-fledged CPS is typically designed as a network of interacting elements with physical inputs and outputs instead of as standalone devices. In this regard, it is critical to define transparent and efficient interfaces between the models. We already started to tackle this challenge and our preliminary studies are partially reported in [111]. By combining the rapid-prototyping approach with the concept of interchangeable interfaces, we are developing a flexible framework for advanced control methods of maritime cranes. The system is based on the use of the Functional Mockup Interface (FMI) [119], which is a tool and independent standard for the exchange of dynamic models and for co-simulation. The framework makes it possible to efficiently integrate different manipulator models, all the corresponding hydraulic systems, various vessels, and the surrounding environment for visualisation. Different control methods can then be transparently implemented and tested. However, considerable work remain to be done concerning the integration and standardisation process. With regard to the possibility of developing alternative and effective control algorithms like the ones presented in Chapter 3, new methods can be implemented and tested as future work. Additional crane models with a larger number of DOFs can be also studied.

With regard to *JOpenShowVar*, the open-source cross-platform communication interface to *Kuka* industrial robots presented in Chapter 4, different control algorithms such as the ones implemented in [82], [120] and [121] may be tested as alternatives to the standard KRC as a topic for future work. Finally, some effort should be put into the standardisation process of *JOpenShowVar* to make it even more reliable for both the industrial and the academic practice.

Considering, the benchmark suite proposed in Chapter 5, new routine tests, different cost functions, and metrics can also be implemented in the future.

Glossary

AHC Active Heave Compensation.
ANN Artificial Neural Network.

BCI Brain-Computer Interface.

DOF In mechanics, the Degree of Freedom of a mechanical system is the number of independent parameters that define its configuration.

EEG Electroencephalography.

FMI Functional Mock-up Interface.
FW Forward Kinematics.

GA Genetic Algorithm.
GUI Graphical User Interface.
GWS Grasp Wrench Space.

IK Inverse Kinematics.
IMU Inertial Measurement Unit.
IR Infra-Red.

KCT Kuka Control Toolbox.
KRL Kuka Robot Language.

LED Light-Emitting Diode.

MRU Motion Reference Unit.

OSC Offshore Simulation Centre AS.

PHC Passive Heave Compensation.
PSO Particle Swarm Optimization.

ROS Robot Operating System.

Bibliography

[1] S. Cobos Guzmán, M. Ferre Perez, and R. Aracil Santonja, "Simplified human hand models based on grasping analysis," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 610–615.

[2] K. Gilpin and D. Rus, "Modular robot systems," *Robotics & Automation Magazine*, vol. 17, no. 3, pp. 38–55, 2010.

[3] M. Yim, D. Duff, and K. Roufas, "Polybot: a modular reconfigurable robot," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, 2000, pp. 514–520.

[4] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. Chirikjian, "Modular self-reconfigurable robot systems [grand challenges of robotics]," *Robotics Automation Magazine, IEEE*, vol. 14, no. 1, pp. 43–52, March 2007.

[5] I.-M. Chen and J. Burdick, "Determining task optimal modular robot assembly configurations," in *Proc. of the IEEE International Conference on Robotics and Automation*, vol. 1, 1995, pp. 132–137.

[6] S. Murata and H. Kurokawa, "Self-Reconfigurable Robots," *IEEE Robotics and Automation Magazine*, pp. 71–78, March 2007.

[7] H. Bojinov, A. Casal, and T. Hogg, "Emergent structures in modular self-reconfigurable robots," in *Proc. of the IEEE International Conference on Robotics and Automation*, vol. 2, 2000, pp. 1734–1741.

[8] C.-H. Yu and R. Nagpal, "Self-adapting modular robotics: A generalized distributed consensus framework," *Proc. of the IEEE International Conference on Robotics and Automation*, vol. 67, no. 1, pp. 1881–1888, May 2009.

[9] H. Lipson and J. B. Pollack, "Automatic design and manufacture of robotic lifeforms," *Nature*, vol. 406, no. 6799, pp. 974–978, 2000.

[10] G. Reshko, M. T. Mason, and I. R. Nourbakhsh, *Rapid prototyping of small robots*. Carnegie Mellon University, The Robotics Institute, 2002.

[11] J. Won, K. J. DeLaurentis, and C. Mavroidis, "Fabrication of a robotic hand using rapid prototyping," in *Proc. of the ASME Mechanisms and Robotics Conference*. Citeseer, 2000, pp. 10–13.

[12] A. M. Dollar and R. D. Howe, "A robust compliant grasper via shape deposition manufacturing," *IEEE/ASME Transactions on Mechatronics*, vol. 11, no. 2, pp. 154–161, 2006.

[13] R. R. Ma, L. U. Odhner, and A. M. Dollar, "A modular, open-source 3d printed underactuated hand," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany*, 2013, pp. 2737–2743.

[14] S. Haidacher, J. Butterfass, M. Fischer, M. Grebenstein, K. Jöhl, K. Kunze, M. Nickl, N. Seitz, and G. Hirzinger, "DLR Hand II: Hard-and software architecture for information processing," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA), Taipei, Taiwan*, vol. 1, 2003, pp. 684–689.

[15] W.-H. Zhu and T. Lamarche, "Modular robot manipulators based on virtual decomposition control," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA), Rome, Italy*, 2007, pp. 2235–2240.

[16] A. R. Murguialday, V. Aggarwal, A. Chatterjee, Y. Cho, R. Rasmussen, B. O'Rourke, S. Acharya, and N. V. Thakor, "Brain-computer interface for a prosthetic hand using local machine control and haptic feedback," in *Proc. of the 10th IEEE International Conference on Rehabilitation Robotics, Noordwijk, Netherlands*, 2007, pp. 609–613.

[17] Y. Li, X. Zhang, Z. Huang, and Q. Hu, "Design of wearable intelligent mind controlled prosthetic hand," in *Proc. of the IEEE International Conference on Mechatronics and Automation (ICMA), Chengdu, China*, 2012, pp. 2462–2466.

[18] J. Yi, N. Yubazaki, and K. Hirota, "Anti-swing and positioning control of overhead traveling crane," *Information Sciences*, vol. 155, no. 1, pp. 19–42, 2003.

[19] P. McKenna and W. Leithead, "Semi-autonomous control of offshore cranes," in *Proc. of the Institution of Engineering and Technology Conference on Autonomous Systems*, 2007, pp. 1–6.

[20] A. Hellrand, L. Moen, and T. Faanes, "Crane control system with active heave compensation and constant tension modes onboard the vessel stena wel/servicer," in *Proc. of the Offshore Technology Conference*, 1990.

[21] F. Sanfilippo, H. P. Hildre, V. Æsøy, H. Zhang, and E. Pedersen, "Flexible modeling and simulation architecture for haptic control of maritime cranes and robotic arms," in *Proc. of the 27th European Conference on Modelling and Simulation (ECMS), Aalesund, Norway*, 2013, pp. 235–242.

[22] F. Nielsen, "Lecture notes in marine operations," *Department of Marine Structures, Norwegian University of Science and Technology, Trondheim, Norway*, 2007.

[23] G. Lebans, K. Wilkie, R. Dubay, D. Crabtree, and T. Edmonds, "Telerobotic shipboard handling system," in *Proc. of the MTS/IEEE OCEANS'97 Conference*, vol. 2, 1997, pp. 1237–1241.

[24] A. Takemoto, K. Yano, T. Miyoshi, and K. Terashima, "Operation assist control system of rotary crane using proposed haptic joystick as man-machine interface," in *Proc. of the 13th IEEE International Workshop on Robot and Human Interactive Communication (ROMAN), 2004*, 2004, pp. 533–538.

[25] M. Schopfer, F. Schmidt, M. Pardowitz, and H. Ritter, "Open source real-time control software for the kuka light weight robot," in *Proc. of the 8th IEEE World Congress on Intelligent Control and Automation (WCICA), Jinan, China*, 2010, pp. 444–449.

[26] G. Schreiber, A. Stemmer, and R. Bischoff, "The fast research interface for the kuka lightweight robot," in *IEEE Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications How to Modify and Enhance Commercial Controllers, Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[27] KUKA Robotics Corporation. (2014, March) "KUKA". [Online]. Available: http://www.kuka-robotics.com/

[28] KUKA, *Expert Programming*. KUKA Robotics Corporation, 2003.

[29] H. Mühe, A. Angerer, A. Hoffmann, and W. Reif, "On reverse-engineering the kuka robot language," in *1st IEEE/RSJ International Workshop on Domain-Specific Languages and models for Robotic systems DSLRob'10, International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[30] KUKA Robotics Corporation. (2014, March) "KUKA, Hub technologies". [Online]. Available: http://www.kuka-robotics.com/en/products/software/hub_technologies/print/start.htm

[31] F. Chinello, S. Scheggi, F. Morbidi, and D. Prattichizzo, "Kuka control toolbox," *IEEE Robotics & Automation Magazine*, vol. 18, no. 4, pp. 69–79, 2011.

[32] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[33] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[34] Y.-G. Zhang, Y.-M. Huang, Y.-Z. Lin, X. Cheng, and F. Gao, "An approach for robot inverse velocity solution using genetic algorithm," in *Proc. of IEEE International Conference on Machine Learning and Cybernetics*, vol. 5, 2004, pp. 2944–2948.

[35] S. Tabandeh, C. M. Clark, and W. Melek, "A genetic algorithm approach to solve for multiple solutions of inverse kinematics using adaptive niching and clustering," *Computer Science and Software Engineering*, p. 63, 2006.

[36] J. Wang and Y. Zhang, "Recurrent neural networks for real-time computation of inverse kinematics of redundant manipulators," *Machine intelligence: quo vadis*, pp. 299–319, 2004.

[37] A. Bouganis and M. Shanahan, "Training a spiking neural network to control a 4-dof robotic arm based on spike timing-dependent plasticity," in *Proc. of the IEEE International Joint Conference on Neural Networks, Barcelona, Spain*, 2010, pp. 1–8.

[38] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[39] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*. Springer, 2010, pp. 760–766.

[40] P. Huang and Y. Xu, "Pso-based time-optimal trajectory planning for space robot with dynamic constraints," in *Proc. of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2006, pp. 1402–1407.

[41] M. T. Das, L. C. Dulger, and G. S. Das, "Robotic applications with particle swarm optimization (pso)," in *Proc. of the IEEE International Conference on Control, Decision and Information Technologies (CoDIT)*, 2013, pp. 160–165.

[42] J. Ni, S. Liu, M. Wang, X. Hu, and Y. Dai, "The simulation research on passive heave compensation system for deep sea mining," in *Proc. of the IEEE International Conference on Mechatronics and Automation (ICMA), Changchun, China*, 2009, pp. 5111–5116.

[43] S. Kuchler, T. Mahl, J. Neupert, K. Schneider, and O. Sawodny, "Active control for an offshore crane using prediction of the vessel's motion," *IEEE/ASME Transactions on Mechatronics*, vol. 16, no. 2, pp. 297–309, 2011.

[44] J. Neupert, T. Mahl, B. Haessig, O. Sawodny, and K. Schneider, "A heave compensation approach for offshore cranes," in *Proc. of the IEEE American Control Conference*, 2008, pp. 538–543.

[45] Y. Chu, F. Sanfilippo, V. Æsøy, and H. Zhang, "An effective heave compensation and anti-sway control approach for offshore hydraulic crane operations," in *Proc. of the IEEE International Conference on Mechatronics and Automation (ICMA), Tianjin, China*, 2014, pp. 1282–1287.

[46] F. A. Leban, J. Diaz-Gonzalez, G. G. Parker, and W. Zhao, "Inverse kinematic control of a dual crane system experiencing base motion," *Transactions on Control Systems Technology*, vol. PP, no. 99, pp. 1–1, 2014.

[47] Z. Masoud, A. Nayfeh, and D. Mook, "Cargo pendulation reduction of ship-mounted cranes," *Nonlinear Dynamics*, vol. 35, no. 3, pp. 299–311, 2004.

[48] Euron - European Robotics Search Network. (2014, September) "Survey and Inventory of Current Efforts in Comparative Robotics Research". [Online]. Available: http://www.robot.uji.es/EURON/en/index.htm

[49] K. Mahelona, J. Glickman, A. Epstein, Z. Brock, M. Siripong, and K. Moyer, "Darpa grand challenge," 2007.

[50] T. Wisspeintner, T. Van Der Zant, L. Iocchi, and S. Schiffer, "Robocup@ home: Scientific competition and benchmarking for domestic service robots," *Interaction Studies*, vol. 10, no. 3, pp. 392–426, 2009.

[51] S. Ulbrich, D. Kappler, T. Asfour, N. Vahrenkamp, A. Bierbaum, M. Przybylski, and R. Dillmann, "The opengrasp benchmarking suite: An environment for the comparative analysis of grasping and dexterous manipulation," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 1761–1767.

[52] R. Vaughan, "Massively multi-robot simulation in stage," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 189–208, 2008.

[53] D. Calisi, L. Iocchi, and D. Nardi, "A unified benchmark framework for autonomous mobile robots and vehicles motion algorithms (movema benchmarks)," in *Proc. of the Workshop on experimental methodology and benchmarking in robotics research (RSS 2008)*, 2008.

[54] S. Moberg, "Robust control of a flexible manipulator arm: A benchmark problem," in *Proc. of the World Congress of the International Federation of Automatic Control (IFAC), Prague, Czech Republic*, 2006.

[55] S. Moberg, S. Gunnarsson, and J. Öhr, "A benchmark problem for robust control of a multivariable nonlinear flexible manipulator," in *Proc. of the World Congress of the International Federation of Automatic Control (IFAC), Seoul, South Korea*, 2008.

[56] D. Prattichizzo, M. Malvezzi, and A. Bicchi, "On motion and force controllability of grasping hands with postural synergies," in *Robotics: Science and Systems*, 2010.

[57] Offshore Simulator Centre AS. (2014, October) "OSC". [Online]. Available: http://www.offsim.no/

[58] C. Ferrari and J. Canny, "Planning optimal grasps," in *Proc. of the IEEE International Conference on Robotics and Automation*, 1992, pp. 2290–2295.

[59] H. Zhang, J. Gonzalez-Gomez, Z. Me, S. Cheng, and J. Zhang, "Development of a low-cost flexible modular robot GZ-I," in *Proc. of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2008, pp. 223–228.

[60] O. Chocron and P. Bidaud, "Genetic design of 3D modular manipulators," *Proc. of the IEEE International Conference on Robotics and Automation*, vol. 1, no. April, pp. 223–228, 1997.

[61] F. Sanfilippo, G. Salvietti, H. Zhang, H. P. Hildre, and D. Prattichizzo, "Efficient modular grasping: an iterative approach," in *Proc. of the 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob), Rome, Italy*, 2012, pp. 1281–1286.

[62] F. Sanfilippo, H. Zhang, K. Y. Pettersen, G. Salvietti, and D. Prattichizzo, "ModGrasp: An open-source rapid-prototyping framework for designing low-cost sensorised modular hands," in *Proc. of the 5th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob), São Paulo, Brazil*, 2014, pp. 951–957.

[63] F. Sanfilippo, H. Zhang, and K. Y. Pettersen, "The new architecture of ModGrasp for mind-controlled low-cost sensorised modular hands," in *Proc. of the IEEE International Conference on Industrial Technology (ICIT2015), Seville, Spain*, 2015, pp. 524–529.

[64] J. Gonzalez-Gomez, H. Zhang, E. Boemo, and J. Zhang, "Locomotion capabilities of a modular robot with eight pitch-yaw-connecting modules," in *Proc. of the 9th Citeseer International Conference on Climbing and Walking Robots (CLAWAR), Brussels, Belgium*, 2006, pp. 12–14.

[65] J. González-Gómez, E. Aguayo, and E. Boemo, "Locomotion of a modular worm-like robot using a fpga-based embedded microblaze soft-processor," in *Climbing and Walking Robots*. Springer, 2005, pp. 869–878.

[66] G. Salvietti, H. Zhang, J. Gonzalez-Gomez, D. Prattichizzo, and J. Zhang, "Task priority grasping and locomotion control of modular robot," in *Proc. of the IEEE International Conference on Robotics and Biomimetics (ROBIO), Guilin, China*, 2009, pp. 1069–1074.

[67] M. Cutkosky, "On grasp choice, grasp models, and the design of hands for manufacturing tasks," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 269–279, 1989.

[68] D. Prattichizzo and J. Trinkle, "Grasping," in *Handbook on Robotics*, S. B. and K. O., Eds. Springer, 2008, pp. 671–700.

[69] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, 2008.

[70] A. Miller and P. Allen, "Graspit! a versatile simulator for robotic grasping," *Robotics & Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.

[71] F. Sanfilippo and K. Y. Pettersen, "OpenMRH: a modular robotic hand generator plugin for OpenRAVE," in *submitted to the Proc. of the IEEE International Conference on Robotics and Biomimetics (ROBIO), Zhuhai, China*, 2015.

[72] C. Goldfeder, P. Allen, C. Lackner, and R. Pelossof, "Grasp planning via decomposition trees," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2007, pp. 10–14.

[73] J. Denavit, "A kinematic notation for lower-pair mechanisms based on matrices," *Trans. of the ASME. Journal of Applied Mechanics*, vol. 22, pp. 215–221, 1955.

[74] OpenGL. (2014, February) The industry's foundation for high performance graphics. [Online]. Available: http://www.opengl.org/

[75] LWJGL. (2014, February) The lightweight java game library (lwjgl). [Online]. Available: http://www.lwjgl.org/

[76] Kahlua. (2014, February) Kahlua, a lua implementation for java. [Online]. Available: https://github.com/krka/kahlua2

[77] Arduino. (2014, November) "Arduino, an open-source electronics prototyping platform". [Online]. Available: http://arduino.cc/

[78] F. Leens, "An introduction to I^2C and *SPI* protocols," *IEEE Instrumentation & Measurement Magazine*, vol. 12, no. 1, pp. 8–13, 2009.

[79] J. Lamothe. (2014, November) "Arduino-Compatible Multi-Threading Library (mthread)". [Online]. Available: http://github.com/jlamothe/mthread

[80] A. Peter. (2013, july) "Efficient Java Matrix Library". [Online]. Available: https://code.google.com/p/efficient-java-matrix-library/

[81] Unity Technologies. (2013, july) "Unity3D". [Online]. Available: http://unity3d.com/

[82] F. Sanfilippo, L. I. Hatledal, H. G. Schaathun, K. Y. Pettersen, and H. Zhang, "A universal control architecture for maritime cranes and robots using genetic algorithms as a possible mapping approach," in *Proc. of the IEEE International Conference on Robotics and Biomimetics (ROBIO), Shenzhen, China*, 2013, pp. 322–327.

[83] F. Sanfilippo, L. I. Hatledal, A. Styve, H. Zhang, and K. Y. Pettersen, "Integrated flexible maritime crane architecture for the offshore simulation centre AS (OSC)," *accepted for publication to the IEEE Journal of Oceanic Engineering*, 2015.

[84] R. M. Murray, Z. Li, S. S. Sastry, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[85] M. Santello, M. Flanders, and J. F. Soechting, "Postural hand synergies for tool use," *The Journal of Neuroscience*, vol. 18, no. 23, pp. 10 105–10 115, 1998.

[86] G. Gioioso, G. Salvietti, M. Malvezzi, and D. Prattichizzo, "Mapping synergies from human to robotic hands with dissimilar kinematics: an approach in the object domain," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 825–837, August 2013.

[87] M. T. Ciocarlie and P. K. Allen, "Hand posture subspaces for dexterous robotic grasping," *The International Journal of Robotics Research*, vol. 28, no. 7, pp. 851–867, 2009.

[88] NeuroSky. (2014, November) "EEG Biosensor Solutions". [Online]. Available: http://neurosky.com/

[89] D. Prattichizzo and J. C. Trinkle, "Grasping," *Springer Handbook of Robotics*, pp. 671–700, 2008.

[90] L. Davis *et al.*, *Handbook of genetic algorithms*. Van Nostrand Reinhold New York, 1991, vol. 115.

[91] A. Lipowski and D. Lipowska, "Roulette-wheel selection via stochastic acceptance," *Physica A: Statistical Mechanics and its Applications*, vol. 391, no. 6, pp. 2193–2196, 2012.

[92] R. Poli, "Analysis of the publications on the applications of particle swarm optimisation," *Journal of Artificial Evolution and Applications*, vol. 2008, p. 3, 2008.

[93] R. C. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," in *Proc. of the IEEE Congress on Evolutionary Computation*, vol. 1, 2001, pp. 94–100.

[94] F. Sanfilippo, L. I. Hatledal, H. Zhang, M. Fago, and K. Y. Pettersen, "JOpenShowVar: an open-source cross-platform communication interface to Kuka robots," in *Proc. of the IEEE International Conference on Information and Automation (ICIA), Hailar, China*, 2014, pp. 1154–1159.

[95] F. Sanfilippo, L. I. Hatledal, H. Zhang, M. Fago, and K. Y. Pettersen, "JOpenShowVar: a flexible communication toolbox for controlling Kuka robots," *submitted to the IEEE Robotics & Automation Magazine*, 2015.

[96] F. Sanfilippo, L. I. Hatledal, H. Zhang, W. Rekdalsbakken, and K. Y. Pettersen, "A wave simulator and active heave compensation framework for demanding offshore crane operations," in *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), Halifax, Nova Scotia, Canada*, 2015, pp. 1588–1593.

[97] B. Porter. (2014, February) "EasyTransfer Arduino library". [Online]. Available: http://www.billporter.info/2011/05/30/easytransfer-arduino-library/

[98] Arduino. (2014, February) "Arduino and Java". [Online]. Available: http://playground.arduino.cc/Interfacing/Java

[99] S. Wang, C. Shi, Y. SHI, and J. CHEN, "Navigation simulator with 360 cylindrical stereo visual system," *Journal of Shanghai Maritime University*, vol. 2, p. 000, 2008.

[100] J. Chen and T. Chen, "Research on standardization of marine simulator training and assessment," in *Proc. of the 3rd IEEE International Conference on Communication Software and Networks (ICCSN), Xi'an, China*, May 2011, pp. 111–114.

[101] Y.-c. JIN and Y. Yin, "Maritime simulators: Convention and technology," *Navigation of China*, vol. 1, p. 002, 2010.

[102] Google Inc. (2014, October) "Protocol Buffers". [Online]. Available: http://developers.google.com/protocol-buffers/

[103] J. Muller, M. Lorenz, F. Geller, A. Zeier, and H. Plattner, "Assessment of communication protocols in the epc network-replacing textual soap and xml with binary google protocol buffers encoding," in *Proc. of the 17Th IEEE International Conference on Industrial Engineering and Engineering Management (IE&EM), Xiamen, China*, 2010, pp. 404–409.

[104] D. Delling, P. Sanders, D. Schultes, and D. Wagner, "Engineering route planning algorithms," in *Algorithmics of large and complex networks*. Springer, 2009, pp. 117–139.

[105] Leap Motion Inc. (2014, March) "The Leap Motion Controller". [Online]. Available: http://www.leapmotion.com/

[106] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2008.

[107] I. Pan, S. Das, and A. Gupta, "Tuning of an optimal fuzzy pid controller with stochastic algorithms for networked control systems with random time delay," *ISA transactions*, vol. 50, no. 1, pp. 28–36, 2011.

[108] W. Bolton, *"Programmable logic controllers"*. Access Online via Elsevier, 2009.

[109] I. Modbus, "Modbus messaging on TCP/IP implementation guide v1. 0a," *North Grafton, Massachusetts (www.modbus.org/specs.php)*, 2004.

[110] PROFIBUS & PROFINET International (PI). (2014, November) "PROFIBUS". [Online]. Available: http://www.profibus.com/

[111] F. Sanfilippo, L. I. Hatledal, Y. Chu, H. Zhang, and K. Y. Pettersen, "A benchmarking framework for control methods of maritime cranes based on the functional mock-up interface," *submitted to the Springer Journal of Marine Science and Technology*, 2015.

[112] R. Suárez, J. Cornella, and M. R. Garzón, *Grasp quality measures*. Institut d'Organització i Control de Sistemes Industrials, 2006.

[113] J. E. Bobrow, B. Martin, G. Sohl, E. Wang, F. C. Park, and J. Kim, "Optimal robot motions for physical criteria," *Journal of Robotic systems*, vol. 18, no. 12, pp. 785–795, 2001.

[114] G. Sarker, G. Myers, T. Williams, D. Goldberg *et al.*, "Comparison of heave-motion compensation systems on scientific ocean drilling ship and their effects on wireline logging data," in *Proc. of the Offshore technology conference*, 2006.

[115] M. R. Lyu *et al.*, *Handbook of software reliability engineering*. IEEE computer society press CA, 1996, vol. 222.

[116] American Petroleum Institute, *Operation and Maintenance of Offshore Cranes, API recommended practice 2D, fifth edition*, 2003.

[117] J. Aleotti and S. Caselli, "Grasp programming by demonstration: A task-based quality measure," in *Proc. of the 17th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2008, pp. 383–388.

[118] E. A. Lee, "Cyber physical systems: Design challenges," in *Proc. of the 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*, 2008, pp. 363–369.

[119] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauß, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold *et al.*, "The Functional Mockup Interface for tool independent exchange of simulation models," in *Proc. of the 8th International Modelica Conference, Dresden*, 2011, pp. 20–22.

[120] F. Sanfilippo, L. I. Hatledal, H. Zhang, and K. Y. Pettersen, "A mapping approach for controlling different maritime cranes and robots using ANN," in *Proc. of the 2014 IEEE International Conference on Mechatronics and Automation (ICMA), Tianjin, China*, 2014, pp. 594–599.

[121] L. I. Hatledal, F. Sanfilippo, and H. Zhang, "JIOP: a java intelligent optimisation and machine learning framework," in *Proc. of the 28th European Conference on Modelling and Simulation (ECMS), Brescia, Italy*, 2014, pp. 101–107.

Secondary Papers

- **F. Sanfilippo** and K. Y. Pettersen, *"OpenMRH: a Modular Robotic Hand Generator Plugin for OpenRAVE"*, submitted to the Proc. of the IEEE International Conference on Robotics and Biomimetics (ROBIO), Zhuhai, China, 2015.
- **F. Sanfilippo**, P. B.T. Weustink and K. Y. Pettersen, *"A Coupling Library for the Force Dimension Haptic Devices and the 20-sim Modelling and Simulation Environment"*, submitted to the Proc. of the 41st Annual Conference of the IEEE Industrial Electronics Society (IECON), Yokohama, Japan, 2015.
- **F. Sanfilippo** and K. Y. Pettersen, *"A Sensor Fusion Wearable Health-Monitoring System with Haptic Feedback"*, submitted to the Proc. of the 11th IEEE International Conference on Innovations in Information Technology (IIT'15), Dubai, United Arab Emirates, 2015.
- **F. Sanfilippo**, L. I. Hatledal and K. Y. Pettersen, *"A Fully-Immersive Hapto-Audio-Visual Framework for Remote Touch"*, submitted to the Proc. of the 11th IEEE International Conference on Innovations in Information Technology (IIT'15), Dubai, United Arab Emirates, 2015.
- L. I. Hatledal, **F. Sanfilippo**, C. Yingguang and H. Zhang, *"A Voxel Based Numerical Method for Computing and Visualizing the Workspace of Offshore Cranes"*, in Proc. of the ASME 34th International Conference on Ocean, Offshore and Arctic Engineering (OMAE), St. John's, Newfoundland, Canada, 2015.
- **F. Sanfilippo** and K. Y. Pettersen, *"Xbee Positioning System with Embedded Haptic Feedback for Dangerous Offshore Operations: a Preliminary Study"*, in Proc. of the OCEANS'15 MTS/IEEE Conference, Genova, Italy, 2015.
- Y. Chu, L. I. Hatledal, **F. Sanfilippo**, V. Æsøy, H. Zhang and H. G. Schaathun, *"Virtual Prototyping System for Maritime Crane Design and Operation Based on Functional Mock-up Interface"*, in Proc. of the OCEANS'15 MTS/IEEE Conference, Genova, Italy, 2015.
- Y. Chu, **F. Sanfilippo**, V. Æsøy and H. Zhang, *"An Effective Heave Compensation and Anti-sway Control Approach for Offshore Hydraulic Crane Operations"*, in Proc. of the IEEE International Conference on Mechatronics and Automation (ICMA), Tianjin, China, 2014, pp. 1282-1287.
- **F. Sanfilippo**, L. I. Hatledal, H. Zhang and K. Y. Pettersen, *"A Mapping Approach for Controlling Different Maritime Cranes and Robots Using ANN"*, in Proc. of the IEEE International Conference on Mechatronics and Automation (ICMA), Tianjin, China, 2014, pp. 594-599.
- L. I. Hatledal, **F. Sanfilippo** and H. Zhang, *"JIOP: a Java Intelligent Optimisation and Machine Learning Framework"*, in Proc. of the 28th European Conference on Modelling and Simulation (ECMS), Brescia, Italy, 2014, pp. 101-107.

- **F. Sanfilippo**, O. L. Osen and S. Alaliyat, *"Recycling a Discarded Robotic Arm for Automation Engineering Education"*, in Proc. of the 28th European Conference on Modelling and Simulation (ECMS), Brescia, Italy, 2014, pp. 81-86.
- S. Alaliyat, H. Yndestad and **F. Sanfilippo**, *"Optimisation of Boids Swarm Model Based on Genetic Algorithm and Particle Swarm Optimisation Algorithm (Comparative Study)"*, in Proc. of the 28th European Conference on Modelling and Simulation (ECMS), Brescia, Italy, 2014, pp. 643-650.
- W. Rekdalsbakken and **F. Sanfilippo**, *"Enhancing Undergraduate Research and Learning Methods on Real-Time Processes by Cooperating with Maritime Industries"*, in Proc. of the 28th European Conference on Modelling and Simulation (ECMS), Brescia, Italy, 2014, pp. 108-114.
- **F. Sanfilippo**, H. P. Hildre, V. Æsøy, H. Zhang and E. Pedersen, *"Flexible Modeling And Simulation Architecture For Haptic Control Of Maritime Cranes And Robotic Arms"*, in Proc. of the 27th European Conference on Modelling and Simulation (ECMS), Aalesund, Norway, 2013, pp. 235-242.
- C. Liu, **F. Sanfilippo**, H. Zhang, H. P. Hildre, C. Liu and S. Bi, *"Locomotion Analysis of a Modular Pentapedal Walking Robot"*, in Proc. of the 26th European Conference on Modelling and Simulation (ECMS), Koblenz, Germany, 2012, pp. 441-447.
- M. J. Callaghan, J. G. Harkin, G. Scibilia, **F. Sanfilippo**, K. McCusker and S. Wilson, *"Experiential based learning in 3D Virtual Worlds: Visualization and data integration in Second Life"*, in Proc. of the Remote Engineering & Virtual Instrumentation (REV) Conference, Düsseldorf, Germany, 2008.

To follow up with my research activity, please visit my website at:
`http://filipposanfilippo.inspitivity.com`.