The walls between art and engineering exist only in our minds.

– Theo Jansen 2006.

University of Alberta

Towards Supervisory Control for Remote Mobile Manipulation: Designing, Building, and Testing a Mobile Telemanipulation Test-Bed

by

Alejandro Hernandez Herdocia

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

Spring 2012
Edmonton, Alberta

To my love, parents, friends, and professors
for all their love, support, encouragement, and teachings.

Abstract

This dissertation has two main contributions, a modular design of a mobile manipulator and a set of tele-operation performance experiments using this platform. To experimentally evaluate system performance and operator preferences, several tests were designed: 1) Which robot camera placement provides the best operator information. 2) A comparison of alternative master-slave motion coordination schemes. 3) A comparison of some semi-autonomous ”software helper” routines to see if they improve manipulation and reduce task load on the operators. Additionally two case studies show how the system was successful in performing complete mobile manipulation tasks, in particular, large-displacement pick-and-place and opening a door to exit a room. A goal of the project was to show how a high-end mobile manipulator can be integrated from off the shelf hardware parts and open-source software libraries.

Acknowledgements

I would like to thank, first and foremost, to my family and friends for always being there supporting and encouraging me to achieve this important step in my academic career and also for the strenuous hours of fun and free-time we spent which helped me to keep a good attitude when the times were adverse. I would also like to thank my supervisor, Martin Jägersand for guidance and insight through out this work and for feedback in this document. I would also like to thank Azad Shademan for his guidance and support.

I would like to thank my fellow students Neil Birkbeck, David Lovi, Adam Rachmielowski, Romeo Tatsambon, and Camilo Perez for their direct and indirect help in this work. Each discussion and collaboration were important keys to achive most of this work. Last, but not least, I would like to thank all the participants of the user studies (you know who you are).

This work was supported by ICT/iCORE Scholarships program.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Robots can deliver accurate, strong, and nimble solutions to many problems. One of these problems is to enable to operate in remote, hazardous, or hostile environments. This interaction with the environment can be done by teleoperation. The basic idea is to remotely issue commands to a robot by means of a user interface.

There are several ways to implement teleoperation [68]. The implementation and the capabilities of the system depend on the sensors, features, and capacities of the robot, as well as on the available interaction devices an operator can use to issue commands and receive feedback from the remote environment. Command devices range from a classic keyboard and mouse to full six degrees of freedom haptic devices or even exact full scale *phantom* replicas of the robot used. Feedback can range from text-based readings of sensors, up to visualizations, remote image display, and force feedback.

Having a variety of different user input/output platforms with different capabilities offers a wide spectrum of possible implementations ranging from very rudimentary (mouse and keyboard with numeric displays) to full immersive (Teledisplay with augmented reality, gesture recognition and haptic feedback). However the latter might seem more appealing, it is more expensive to implement and it is only desirable when teleoperating a robot which will function as a surrogate teleoperator. If a setting that will feature some level of autonomy is desired, this telepresence is not crucial. If the user of a system wants to "tell" the robot to perform an action in a more descriptive way rather than in an exact replicated manner, it is preferable to improve the user interface to ease issuing commands and relieving the operator from unnecessary task load.

Figure 1.1 shows a classic setting for teleoperation. Teleoperation architectures have three important components:

Master Device Provides the commands for the system. This device will also render the

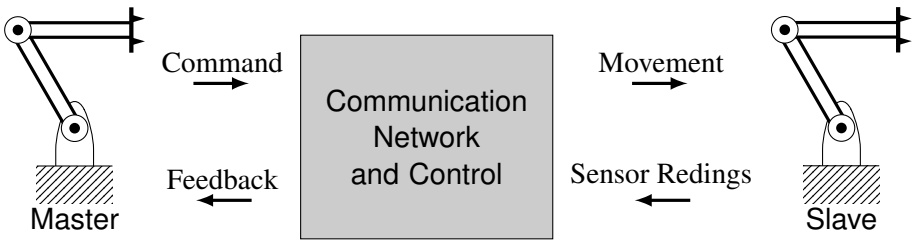


Figure 1.1: This is a classical teleoperation seeting with haptic feedback consisting of a master device (left), the controller/communication module (center), and the slave side. The master side is usually where the operator issues commands and the slave side is the one in contact with the environment

feedback to the operator.

Slave Device Is the remote robot which is performing the commanded movements. This node is likely to have colision, force, and torque sensors to provide the feedback information to the master.

Network and control This is the means of communication and together with a set of algorithms that pair the Master and the Slave.

In a normal setting for teleoperation it is natural to include cameras for visual feedback. This allows the operator to see what is happening in the remote environment. Including vision sensors provides the operator access to a large amount of information. Not only it is possible to retrieve high fidelity images from the remote environment, but it is even possible to reconstruct a model of the remote site by using the information from the visual sensor [64].

Teleoperation has several shortcomings. One of these is the delay [67] that is inherent to the system because of the master and slave stations being afar. One way to deal with the delay is by using Predictive Display (PD). PD uses a reconstructed virtual environment which is generated based on the visual information received. Then, this information is presented to the operator in a "predicted" model of the remote environment. This feature allows the operator to become oblivious to the delay in the system.

Another approach is to use control theory and make the system passive by giving up some transparency. The technique consists of making the communication network passive (dampening and not producing any energy). This renders the system usable and stable, at the cost of dampened motions.

One of the goals of robotics is to achieve full autonomy[1] but this is yet in exploratory

[1] Autonomy comparable to that of human skills (communication, physical strength, etc.

state. A way to approach this promised robotic autonomous behavior is to build basic modules that could be used as building blocks to gradually achieve more complex super-modules. These basic routines automate a very particular behavior, thus, they need to be defined in the most simple ways to improve their future usability. Defining and designing software helper routines or "helpers" would provide a base towards gradually developing more autonomous and less explicit teleoperation. These aiding software routines would encode commonly used behaviors such as repetitions or restrictions which delimit and define a task. For example, sequencing such tasks and letting the robot execute a playlist of those helpers, the operator can complete a full activity with minimal specification from the user, thus removing task load.

Due to the interdisciplinary nature of robotics, this work has some overlap, not only in several topics of computing science (*e.g.,* parallel and distributed systems, software engineering and computer architecture) but also in several areas of engineering. In fact Robotics itself is of a mechatronic nature, meaning that it overlaps computing science, control theory, mechanics, and electronics. Although the focus of this work is set on computing science, at times it becomes necessary to include a full mechatronic focus to gain a better perspective of the issues that concern this study. Figure 1.2 shows a mindmap of the related areas to this work.

1.1 Motivation

Nowadays teleoperated systems are to some extent real and deployed. Example is the Da Vinci [28, 31] Surgical System for tele-surgery applications and the Mars Exploration Rovers [54]. These systems have different degrees of autonomy. The former has a less "autonomous" approach and the latter exhibits a more supervisory control approach. This difference resides in their application.

The Da Vinci Surgical System is meant to perform laparoscopic surgery with delays no more than the latency of the system in order to allow the operator to make fast decisions. The operator, a surgeon, performs a *risky* task, an operation in a human. If the system becomes unresponsive or too delayed, this may cause the patient's life.

On the other hand, the Mars Exploration Rovers are designed to operate with supervisory/autonomous control by specifying the task in a more abstract way. The delay that this system faces is too large that it is necessary to plan the activities and send abstract messages to tell the robot what to do. In this case, the decisions are not taken at the time of operation

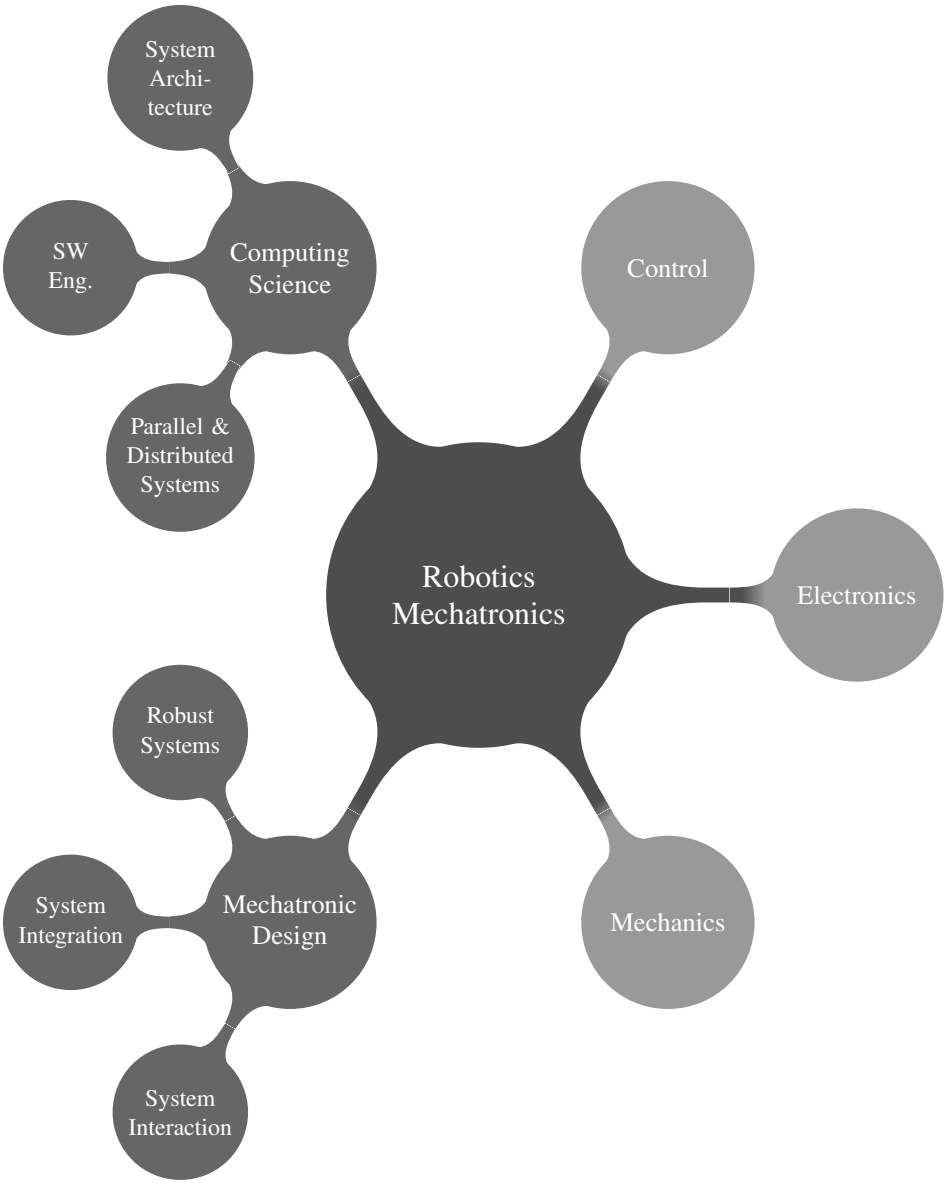


Figure 1.2: The nature of the project: This work has some overlay not only in topics of computing science but also in several areas of engineering.

but before relaying the commands to the robot. Although an error might have a high cost, the loss is only material.

However, none of the current approaches consider an intermediate solution less taxing on the operator[2] and more closely supervised while relying less in the robot's autonomy. The major motivation to consider a bottom-up approach towards attaining autonomy is that in larger kinematic chains (mobile manipulators for instance), autonomous behavior is still in its early stages of development and the complexity of the problems to solve makes them difficult to tackle [26].

Teleoperation can be "reliable" because there is a human in the loop but the onus put on the operator due to delay [9, 10] and interfaces put makes it sometimes difficult to use. On the other hand, autonomy although difficult to attain, it provides ease of command. Neither teleoperation or autonomy alone seem to provide a robust solution to general manipulation. By using the best of both approaches, and finding some middle ground between them it is possible to obtain more reliable and robust supervised semi-autonomous behaviors. By **developing a teleoperation framework that sets the bases to perform teleoperation tasks**, it is possible build a set of building blocks. Later, these building blocks can be used to develop more complex routines and behaviors. This will be detailed in a chapter to follow.

Teleoperated systems have to be designed with a mechatronic systems methodology rather than focusing on a single approach (mechanics, electronics, control, or computing systems). As part of this work, another **contribution** is presented: **a methodology** that will benefit the **development** of teleoperated mobile manipulator systems.

Up to now, in robotics it is particularly common to see two scenarios:

- *Ad hoc* settings to deploy the proof of concept, but failing to operate robustly
- Custom Engineered Systems that take years to develop using too many resources which sometimes become outdated in terms of computational power by the time they are ready to deploy.

Although these two scenarios are becoming less common in practicum, they are still the basic approaches taken by the robotics community to developing a system.

A favorable design has not only to be a test-bed for a single demonstration of a proof of concept, but also provide a framework that allows several other proofs to be performed, and

[2] In terms of issuing commands or planning ahead

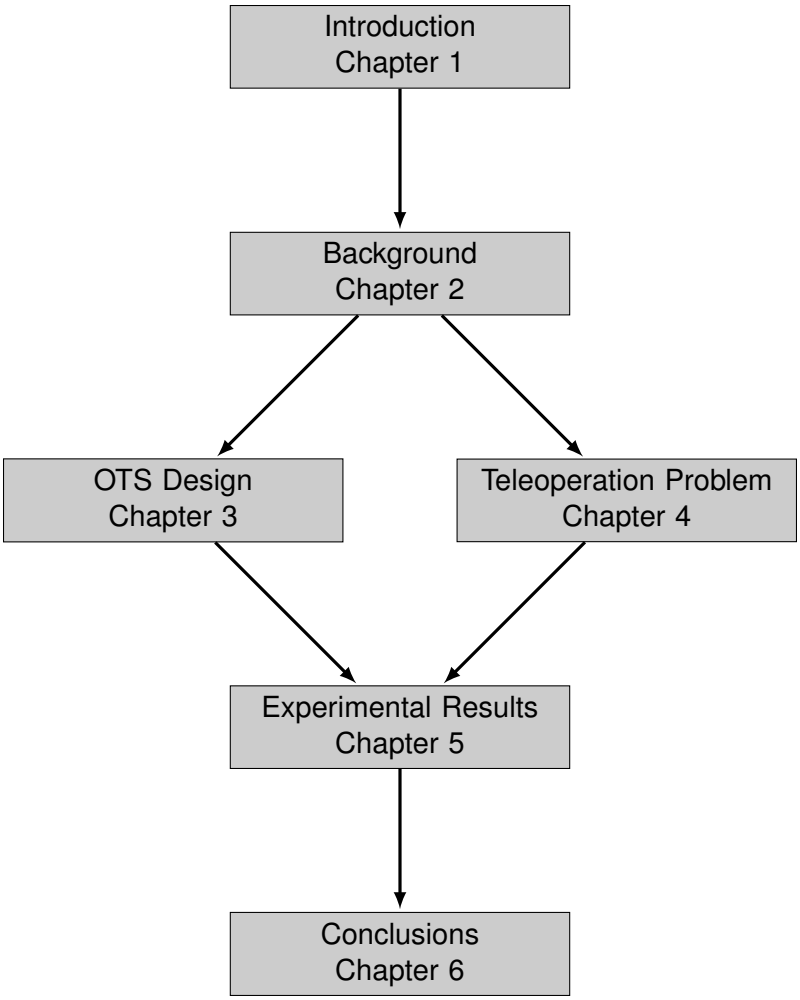


Figure 1.3: Roadmap/Outline of the Thesis

can even be used as a prototyping scenario. It follows that the possibility of upgrade has to be considered, as the system will need to be updated and customized.

1.2 Thesis Outline

Figure 1.3 shows an explicit roadmap of this document. It provides a guide on how to read this dissertation.

In Chapter 2 the findings from the literature review, which informed and motivated this research, are presented along with some related theoretical background. This is immediately followed by two chapter which can be independently read:

- A methodology for designing teleoperated mobile manipulation systems is presented in Chapter 3, along with the implemented system used for the experiments. This

implementation and the methodology were presented as a conference paper [40].

- A framework to generate the teleoperation software helper routines is described in Chapter 4. The implementation of the software helper routines is explained in this chapter as well.

Although it might seem unnatural to first implement the system rather than developing the proof of the concept at hand, it is a better course of action to develop a working system and then continue to add modules as further studies require. This allows developing a more generic testbed that can be used later for other studies. It is better to solve the inherent challenges of designing and building a complex multi-module system earlier in the development and leave the issues of adding application-specific modules for later.

Experimental results, being the **main contribution** of this work, will be presented in Chapter 5. These results serve as an integration point for the ideas presented in Chapters3 and 4. The experiments studied some detailed aspects: 1) Which camera positioning provide best information to perform a task. 2) The use of alternative master-slave motion coordination schemes. 3) A comparison of some semi-autonomous ”software helper” routines to see if they improve manipulation and reduce task load on the operators. Two case studies show how the system was successful in performing complete mobile manipulation tasks, in particular, large-displacement pick-and-place and opening a door to exit a room.

Finally, a summary and a discussion of future work are presented in Chapter 6 to conclude.

Chapter 2

Background and Related Work

This chapter will review some concepts and the most relevant related work. It will also provide a comparison including some of the robotic systems for teleoperation and telepresence as well as haptics and the techniques for dealing with time-delay. First, a survey over the main related problems of controlling robots is detailed in Section 2.1. Then, an overview some background work on haptics related to teleoperation is presented in Section 2.2. Section 2.3 surveys some of the existent systems for mobile manipulation and teleoperation. A short discussion concludes this chapter in Section 2.4.

2.1 Robotics: From Teleoperation to Autonomy

For some time now, teleoperation has been around as a means of controlling robots. Everyone to some extent has been in contact with such an approach when operating something from afar, *e.g.,* a radio control toy. However, in most of these cases teleoperation has no particular purpose other than entertaining or providing a comfortable way to perform an action[1].

The uses of teleoperated systems are plenty because such a system isolates the operator by putting some physical distance between the device and the person thus physically decoupling and protecting him or her. Teleoperated systems can be useful when interacting with some dangerous and/or otherwise unreachable environments or things, *e.g.,* defusing a bomb, performing extra-vehicular-activities in space, exploring other planets, or dive to the bottom of the sea.

However, there is an important trade-off when putting distance between the operator and the teleoperated device: time delay [34]. Having distances of no more than a couple of tens of meters and low latency systems, time delay might appear negligible. The problem

[1] If one considers the remote starter for a car or the remote for the TV set like a teleoperation device.

arises when larger distances are introduced and the communication channel between the operator and the device is not transparent due to latencies in the system and the communication channel. Whoever has played on-line or networked games has experimented in some extent the effects of time delay or lag which sometimes renders the game unplayable and frustrating.

If time-delay and the need of a more "intelligent"[2] interaction with the devices are considered, then the true need of **semi-autonomous supervisory control** arises. The level of autonomy of a device can be described as the amount of interaction and description a system needs to successfully perform a task. The more description/interaction it needs, the less autonomous the device is. Figure 2.1 shows the basic configurations of teleoperation and autonomous systems.

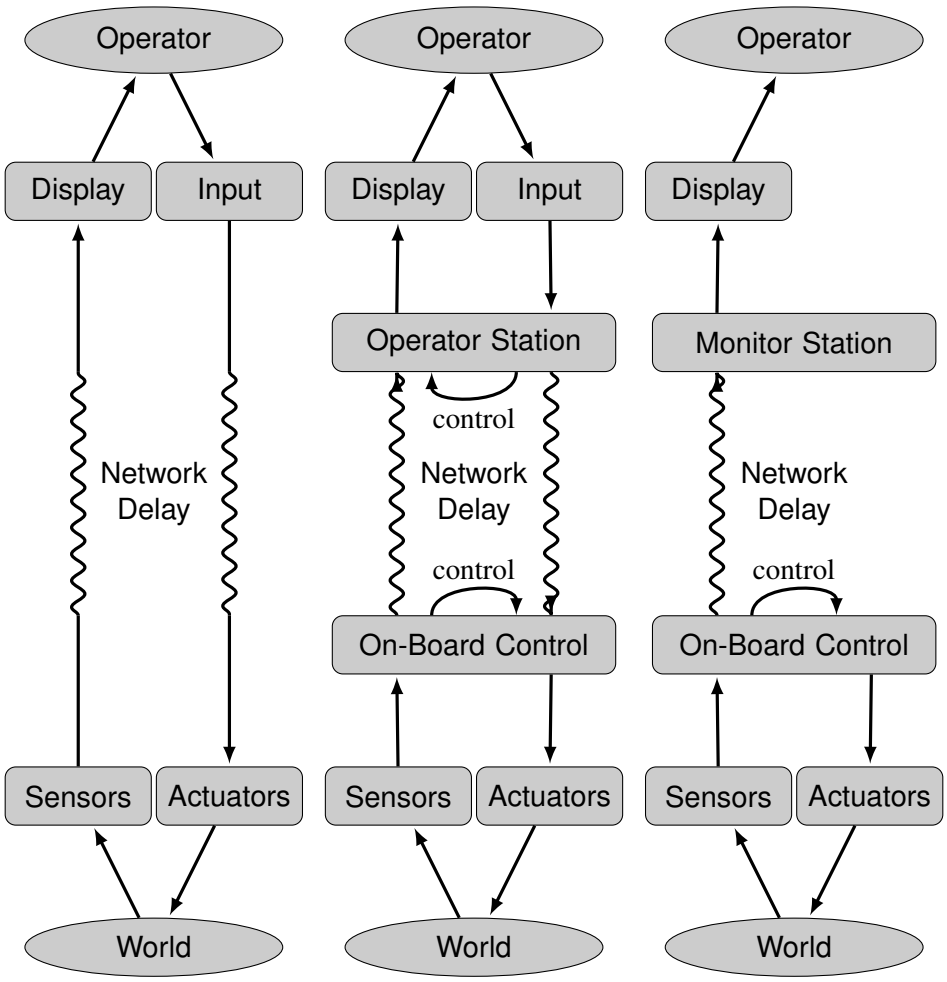
This work focuses in teleoperated mobile manipulators. These may or may not have certain low level autonomy already implemented. For simplicity, those cases where autonomy is needed as a direct interaction asset, *i.e.,* when communicating and operating with a robot as if it were a peer, are left undiscussed; these latter are the ultimate target of future works along this line of research. Given these considerations, some concepts used through this work can be defined.

2.1.1 Teleoperation, Telemanipulation, and Telepresence

As mentioned in Chapter 1, there are several ways to issue commands to a remotely operated robot, several ways in which the robot's sensors can be displayed to the operator, and several possible actions the robot can perform in the environment. Depending on the different characteristics of a system and its capabilities, a system can be classified as being generally teleoperated, capable of telemanipulation, and/or capable of telepresence.

- **Teleoperation** A system is teleoperated when the operator issues commands to a remote device far away from the control station. Teleoperation may include a remote display which provides the operator with the readings taken by the sensors in the remote device. In plain teleoperation explicit orders are issued to the device's actuators and remote sensor readings are displayed to the operator. In the case of *Manual Control*, the operator is closing all the control loops. In *Supervisory Control* some/most of the lower level, and even some higher level, control loops are closed by a processing unit on board the teleoperated robot, leaving the most abstract control loops for the user. *Full Autonomy* leaves the user as a witness and only abstract

[2]Do more with less explicit descriptions or explanations



(a) Teleoperation (b) Supervisory Control (c) Autonomous Control

Figure 2.1: This figure shows the basic three configurations ranging from plain teleoperation (left) to full autonomy (right). (a) For teleoperation the control loop is closed by the operator. (b) Supervisory control relies in both the operator and a computer to close control loops. It is usual to leave the lower level control loops for the computers while the operator controls the higher level, more abstract control loops. (c) Under full autonomy, the onboard computer (and in a few cases an off-board computer) is responsible for doing this, while the operator is only given the information and a high level interface to monitor the system.

commands/objectives/goals are passed to the system. In this latter case the system takes care of closing the control loops and performing several tasks without supervision [67, 68, 69].

- **Telemanipulation** Teleoperated robots which can manipulate objects on the remote environment are telemanipulators. Any teleoperated robot capable of dexterously interacting with the remote environment can be considered a telemanipulation device. A explicit differentiation between Teleoperation and Telemanipulation is necessary as the former encompasses a broader variety of systems, while the latter specifically refers to those capable of *manipulation*.

- **Telepresence** A teleoperated system having a transparent communication channel and an interface that allows the user to "feel" in the place of the robot at the remote scene, is a telepresence system. Such systems enable the operator to seamlessly interact with the remote environment. This feature is one of the most desired on pure teleoperation settings as it allows for an immersive experience for the user. When no significant delays are present, it can be obtained by using augmented reality and natural ergonomic control interfaces. When time delay is present, this can be achieved by Predictive Display and Virtual-Reality models. Telepresence needs an increase of the remote sensory information to generate a more complete reconstruction of the remote scene.

2.1.2 Relaying Information from the Remote Scene to the Operator

Most of the times in robotics applications it is necessary to develop a way to represent the environment. In teleoperation, the main concern of environment modeling is not focused on how to abstract the world and process the sensor readings, but how to synthesize data and construct a condensed model to ease the job of the operator. Which sensors, what and how much information can these sensors provide needs to be taken into consideration to design the preprocessing and the relay of information. The communication channel, being a finite resource, needs to be used efficiently while still providing the most relevant information[3]. For example, if modeled correctly, the information received by a camera can be used first to generate a coarse geometry of the environment [47] and then used to generate visual goals to do visual servo control over the robot. Although environment modeling is not a crucial

[3]This is particularly necessary when including vision sensors as they provide rich information but the bandwidth needed to transmit it could be too large

aspect in teleoperation, it should not be disregarded as it determines how difficult to use is the user interface.

Motion mapping and data display are most of the times obviated and defaulted as the operator is assumed to be very adaptable to whatever information is given to process. Although true, teleoperation is largely facilitated if the information is formatted and coded into various types of feedback, *e.g.,* changing colors of the display, stiffening the input device, sound warnings, etc. This improves the performance of the teleoperator [69] by using other available channels of interaction between the machine and the human.

Since sensors cannot provide full exhaustive information of the remote scene, it is crucial to take advantage of any available information, filtering or combining it to generate new useful information. Although having more information might yield better decisions, operators can only pay attention to a handful of gages and readings. Thus, designing "clever" new features is necessary to ease making sense of what is happening in the remote scene.

2.2 Haptics

Haptic feedback is, in a strict definition, feedback through touch and contact. Anything that is perceived through touch and expressed as forces, vibrations, movements, etc., is haptic feedback. If the definition is extended to truly include everything that can be perceived through touch, then it is necessary to include temperature, textures, stiffness, and hardness. These are all properties that could be modeled and output by a haptic device yet not all of them are as easy to implement.

Haptics has been around for quite a time. The first uses were for handling radioactive materials. Nowadays haptics is used widely in several applications other than just for teleoperation as they are used extensively in Virtual Reality, Video Games, Simulators, and Mobile Consumer technologies. In teleoperation, most of the times, the setting is designed such that the position is sent from the master side to control the torque exerted by the actuators on the slave device. The position on the slave device, in turn is used to control the torque output on the master side.

2.2.1 Common Uses of Haptics

Haptic devices are usually used in research and applications on telesurgery [48, 49], manipulators, mobile robots [80], mechanical and graphic design, and virtual reality [61]. Efforts are divided into those applications dealing with virtual and real worlds. In the former, most of the times there is no significant delay in the issue-execution of a command or in the

interaction-feedback provided by the environment. However, there have been some considerations of working with delay in Virtual Reality scenarios [37]. Real world applications deal with more complex configurations where haptic feedback is done with respect of actual readings from a robot. In this scenario latency is unavoidable due to the of complexity of such systems and the physical distance between the command center and the teleoperation device.

Most systems dealing with real world applications of teleoperation with haptic feedback privilege the use of two very similar architectures and geometries for the haptic interface and the manipulator. This reduces the design burden and the need for motion coordination. An important downside of this approach is the cost of such a configuration.

2.2.2 Problems With Current Approaches

As mentioned before, having two similar devices for the master and slave sides might be costly but convenient. On the other hand, having a (smaller) master with different geometry than the slave's might be cheaper but introduces some complications which have to be dealt with:

Motion Coordination The differences of the kinematic chains of the master and the slave, demand that a mapping between the motions of the master and the slave is found. This problem is present unless the master is a scaled replica of the slave.

Scale of Motion It is desirable to have different scales available to command movement from the master to the slave. This feature allows fine and coarse manipulation.

Scale of Force Related to the scale of motion, scaling the forces is another quandary that has to be taken into consideration. Forces from the slave side might not be within the capabilities of the master. Moreover, when remotely manipulating fragile objects it would be desirable to have enhanced force reflection.

Differences in the scale of motion and force could potentially expand the capabilities of the teleoperated robot. Having a system which can operate both roughly or softly and in big or small scale movements could very well improve the types of tasks that can be performed.

A number of interfaces have been proposed for teleoperation and telepresence. The main goal of all the works is to make the interaction as natural as possible, either by providing a natural way to interact or command the robot [57, 58], or to make the control an immersive experience [80, 25, 24]. Although these works have outstanding user interfaces,

one lacks tactile feedback for the user and assumes some degree of autonomy while the other was developed using a large budget.

The Control point of view of Teleoperation

The major goal of most of the approaches is guarantee that the network is transparent and passive. These two concepts are very related and widely used as parameters to characterize a teleoperation system. As in every application, there are some issues that the designer has to deal with. Tuning the system to work with time delays demands trading off transparency for passivity. A system that is very transparent might be very unstable, thus non-passive, while a very passive system might not be transparent [75, 76].

Passivity is the condition of the communications channel of a teleoperation system to damp and not add any energy into the system. In more concrete terms, a system is passive when the communication network dissipates the energy entering the communication channels.

Transparency is the property of the system to provide exact feedback to the user while executing a command on the slave device as close as possible to the way it is commanded in the master device. A transparent system allows for some degree of telepresence. An operator should be able to feel exactly what is happening in the slave side without any fuzziness[13].

For bilateral teleoperation, two of the most important architectures are wave variables and the 4-channel architecture. A nice comparison between these two approaches is presented by Christiansson in [27]. Some improvements, like the one presented by Ott and Nakamura in [56], have been presented for distributed control robots, such as mobile manipulators. In this latter case the paper presents how the four channel architecture can be used to communicate between "collaborating" distributed robots.

A control diagram of a teleoperation system implementing the 4-channel architecture can be seen in Figure 2.2. Note the many parameters that have to be tuned for the controllers on each side and the communication channel itself. Each darkened box (C_m, C_s,C_{1to6}) stands for a filter which has to be selected in such a way to guarantee passivity of the controller. Moreover, the operator's (Z_h) and environment's (Z_e) impedances are difficult to obtain and vary from different operators and environments.

Most of the control approaches focus on improving the passivity of the network. They are formulated considering that the delay between the master and the slave is constant.

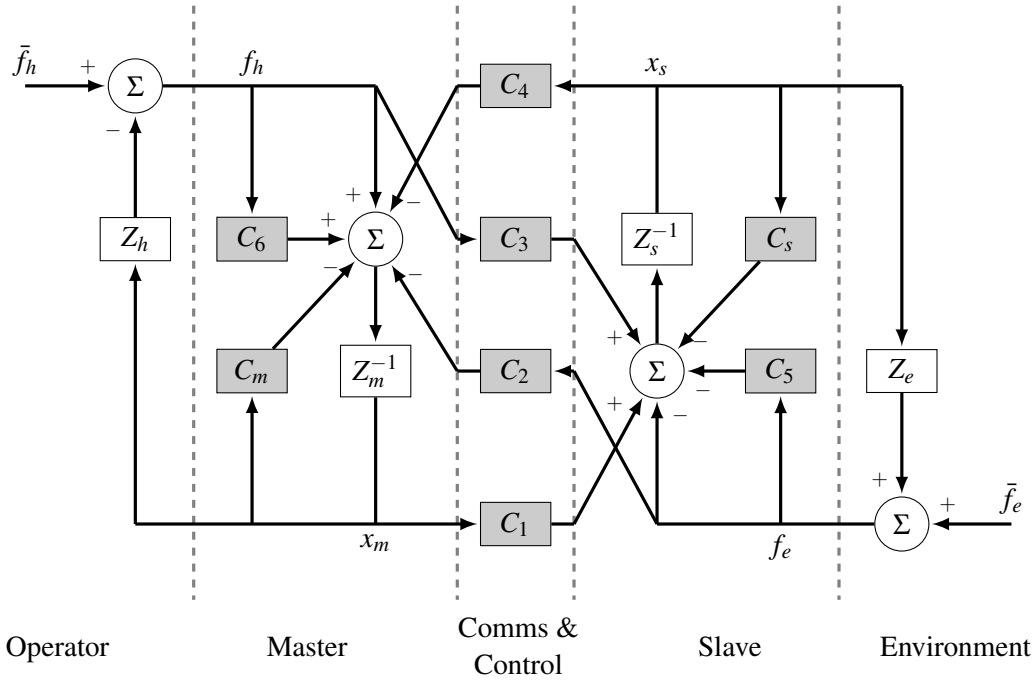


Figure 2.2: The 4-Channel Architecture. C_{1to6}, C_m, and C_s are filters which have to be tunned. Z_h is the operator's impedance. Z_e is the impedance of the environment. Z_m and Z_s represent the master's and slave's characteristics. f_h and f_e are the forces input to the system by the operator and by the environment respectively. x_m and x_s are the master's and slave's rates.

In theory the delay can be infinite yet in practice it cannot be longer than a couple of seconds [47]. When the delay is in the order of tens of seconds, the system becomes so stiff due to the dampening in the communication channel, that the operator cannot effectively command the system to move. Apart from the system becoming gradually difficult to move, the operator also starts feeling the degradation of the feedback [79]. When dealing with variable delay [20, 83], such as in the case of telemanipulation over internet, the chances of the system becoming unstable increase [83]. This is because variable delays are difficult to model.

Most of the works done up to date focus either on the control point of view of the teleoperation problem, or on the task load posed to the operator, particularly in how time delay affects teleoperation [9, 10]. Some other amount of works focus on how to effectively use haptic and visual feedback, like in the case of Yip *et al.* [82]. Although time delay and haptics are perhaps the most important problems to address, some other topics need to be treated in parallel: asymmetric interfaces for mobile manipulation, system design and prototyping, and intuitive/ergonomic controller schemes.

2.3 Existing Robotic Systems for Teleoperation

Having a suitable test-bed can be a great help to test concepts, design new algorithms, and perform experiments. Sometimes it is enough to have a simulator, but most of the times a real device helps roboticists gain insight towards the adjacent problems: Hardware-Software interaction, System Integration, System Complexity, Robustness, etc.

Mobile robots, research and low-end arm manipulators have become, to some extent, inexpensive. The availability of such resources allows a greater audience to get involved in research of manipulation, mobile robotics, and mobile manipulation. Previously, most of the mobile and static manipulation systems had to be custom engineered because of certain limitations such as payload, mobility, dimensions and capabilities. Such complex systems benefit from a more integral design where all variables are taken into consideration at the same time but the paradigm is resource-expensive. Even nowadays it can be argued that the performance of a system is better when custom engineered, but the commercially available options allow to produce systems which are not too far behind. A problem with using custom engineered platforms is that it limits the community to participate as the platform is not as accessible as a commercial out-of-the-box or a modular system.

Through the literature, mobile and static manipulators are described in works about humanoid robots, human-robot interaction, rehabilitation, assistance and space exploration. The next few subsections overview some of the relevant works. The following classification is based upon the type of approach taken to developing each of the systems. The included projects are not an exhaustive list of all the mobile manipulators as it only includes the most relevant and well-known works in the field.

2.3.1 *Ad Hoc* Systems

Most of the mobile telemanipulation systems being deployed currently fall into this category. These systems have the advantages of a long refining process and a low-level tunning capability. These advantages come at the cost of a resource-expensive development and low-audience accessibility. Moreover, they do not offer the alternative of seamless integration with already developed algorithms because these systems have some characteristics that make it hard to import any software which was not explicitly developed for the platform. The same holds true for exporting software.

This cageory includes NASA's Robonaut [8, 15, 16, 33, 52, 65] which is already Deployed in the International Space Station [2], Georgia Tech's Golem Krang [72], University

of Massachusetts, Amherst's UBot-5 [29, 30, 77], Karlsruhe Institue of Technology's Armar [3, 4, 5, 6, 7, 11, 12], DLR's Rolin Justin [17, 18, 44, 55], and Waseda University's Twendy [53, 74]. All these systems have taken at least a decade to develope from the first prototypes to the stage in which they currently are. They feature interesting integration and outstanding mechanical capabilities, but in most cases the control is still in the low- and mid-level range of abstraction. Most of the tasks should be described to the robot with very explicit instructions.

Most of these platforms, although flexible by nature, are application specific. For example, although Robonaut is capable of using human-scale tools, it would not be cost efective to deploy it in a house setting. Despite being very promising platforms for developing algorithms, their setting is too specific to be able to port the generated software. A software generalization process would be needed to enable the software to be generic enough to be used in other platforms. Therefore, not that many researchers can benefit from the developed code.

2.3.2 Hybrid Systems

Given the fact that most of the enthusiasts interested in doing research in telemanipulation are not specialists on every technology needed to develop a full-scale telemanipulator, a good option is to rely on commercially available parts to build up a system. Although the components are available, there's still a need to engineer the system in such a way that it meets certain specification related to the application. Moreover, in some cases available components cannot provide what is needed which has led some research groups to design hybrid systems. These still rely in a custom engineering development of the system but also in the use of commercially available modules. Such systems take less time to develop than their *Ad Hoc* counterparts, but most of the times they still have the problems of software portability. Particular hardware configuration and application specific configurations make importing and exporting difficult. A great example in this category is University of Massachusetts Amherst's UMan [42]. This mobile manipulator features a custom-made mobile base and a Barrett WAM for manipulation.

Despite there is available software for the commercially available parts, there might be little or no software usable for custom-engineered parts. Although possible to overcome the need of designing and implementing software and hardware for parts of the robot, there is still some parts that have to be developed.

Another example is MIT's Cardea [51, 21]. Which uses the well-known Segway RMP

mobile base and a custom made arm manipulator. As UMAN and Cardea, there are several other examples which fall in this category. It is even arguable that all systems fall within this category as all systems rely on previously developed modules which are commercially available.

2.3.3 Commercial-Off-The-Shelf Component Integration

Nowadays less expensive, bulky and accurate sensors are available making it easier to select modules to integrate into a robot; it is possible to find inexpensive industrial grade fast cameras, very accurate range sensors using laser technology, IMUs, high-payload mobile bases, nimble robotic manipulators without a separate amplifier cabinet, and commercial haptic interfaces to prototype a telemanipulation system without spending ten years of development. The various available modules makes designing and integrating a mobile manipulator (and robots in general) a much easier task if the requirements are not tight or demanding. Given that there are many modules commercially available, it becomes less cumbersome to prototype a test-bed to develop algorithms for robotics. Available sensor, actuator, computational, and robotic modules enable rapid and agile prototyping of teleoperation systems.

Having commercially available products means that a community is potentially using and developing software for such modules. This is beneficial as there are better chances to benefit from and even use already developed software. Not only the software is available but it has been potentially tested several times. The existence of a user/developer community distributes the overall workload and resources needed for developing complex systems. Off the Shelf Component Integration allows researchers to focus in the high-level algorithms as the paradigm reduces the time spent developing low-level interfaces and drivers. However, the advantages of having a supporting community and a ready to use modules come at the price of partial ability to modify these modules. The main risks are lack of support for certain configurations, incompatibility and inconvenient integration/interaction.

Up to date, there are only a few teleoperation and mobile manipulation systems which are integrated from Commercial Off The Shelf components. It was only recently that robotic manipulators became available in more compact settings and also offering open-software architecture. There are a few examples of robots using this paradigm: University of Massachusetts Amherst's Dexter [78], Intel Research's HERB [71, 55] and the arm developed at the Royal Institute of Technology in Stockholm [70].

2.3.4 Ready-out-of-the-box Systems

Lately some mobile manipulators have become commercially available as ready to use systems. These robots present a good alternative to start doing research without major problems of designing a system. Despite appearing attractive, their price is substantial and usually a certain amount of time is needed to learn the basics of their operation. Most these systems have been around for a little more than half a decade [1], while there are some other cases which are very recent such as Willow Garage's PR2 [81].

These systems provide proven test-beds, fully integrated systems, and working software on the platform. Users only have to allocate time to learn the higher level interfaces. Although good options to start with, these platforms cannot be upgraded that easily. Incorporating or updating with new hardware becomes cumbersome or expensive. Moreover, special attention has to be put towards the implications of doing these upgrades and how will they affect the overall system's performance.

2.4 Discussion

Robotics promises to provide answers to repetitive, cumbersome, and dangerous tasks. While full automation is still work in progress, plain teleoperation and explicit programming are sometimes too unfeasible because of the onus put on the operator. Supervisory control is an option to these two alternatives as it takes the best of both worlds. Moreover the supervisory control paradigm can be used to build up on gradually more autonomous behaviors for general purpose robots.

An important aspect of teleoperation is how to command and retrieve information from the remote robot. Haptic feedback can be effectively used to provide the user with better and more information of the remote scene. Teleoperation, from the point of view of control, focuses on the development of transparent and passive interfaces, yet other aspects need to be taken care of, such as the ergonomics of the input devices and the control schemes used to command robots.

Designing and building a telemanipulation system can become difficult and complex due to the large number of modules involved. It is also a challenging project because of the inherent complexity of such systems. The high number of variables involved due to the robot, the time delay between the issue and execution of commands as well as limited sensory information availability make it an interesting and complex problem to study as its applications can serve largely in several tasks deemed dangerous and risky for a human to

perform.

Several systems have been developed. *Ad Hoc*, Hybrid, Off The Shelf Component Integrated, and Ready Out of the Box systems have been built yet the latter two seem to be the best options nowadays for the community interested in mobile manipulation. They provide ways to focus on developing the algorithms and portability of the software because they provide proven and supported technologies.

Chapter 3

Off-The-Shelf-Component System Design and Integration

Throughout the robotics literature one can identify several mobile manipulators which have been custom designed and engineered in labs worldwide. These "one-of-a-kind" have very particular configurations and applications which makes the research produced by these labs very difficult to port and test by others. The algorithms proven to be successful for a particular kind of robot will have to be tunned and adapted before they can be used in another platform. Mobile manipulators could become more prevalent if designed and assembled from commercially available components instead of being locally engineered and built [40]. This chapter presents the methodology and the integration steps taken while putting together a mobile manipulator.

The main ideas kept through the development of the system were **modularity**, **flexibility**, and **expandability**.

Modularity A system is modular when the parts of a system have defined inputs, outputs, and functions. Modularity makes it possible to substitute a part by another doing the same function and having the same inputs and outputs (only varying the way the module performs its function). The system therefore remains as "able" as before the substitution as the whole system's behavior is not modified on the general outcome. The substitution of modules has the purpose of improving the efficiency and/or the performance of the system in general.

Flexibility A system is flexible when it can be used for multiple purposes while keeping reconfiguration minimal. Flexibility is a main characteristic of multipurpose systems such as mobile manipulators. This feature makes a system capable of hosting various projects without having to incur in major changes to its architecture or overall

configuration.

Expandability If a system is expandable then it can be updated and modified to satisfy future improvements and functionalities. When new functions can be added into a system, its productive life increases. New useful components can be added to satisfy new research goals.

A system featuring these three characteristics can host many future research projects because it is not bound to a particular application and it can be updated and extended as needed.

Off-The-Shelf-Component integration provides an alternative to the classic approaches of system development. The designer trades off some of the ability to control details in the development, capabilities of the resulting system, and understanding of the low-level processes to gain agility in the overall design process. Another benefit of this paradigm is that expertise on every area related to the project is no longer required. Moreover, the developer becomes an active member of a community developing and testing the components.

The paradigm has some risks:

Technical Refers to the ability of adopting the knowledge of the technology and understanding a component. It also includes the limitations imposed to the design because of the lack of an exact component that fulfills a particular function.

Operational This risk refers to the operation of the full system being within specification or not. Sometimes, particular behaviors of the system are obviated which could decimate the overall performance of the system in particular scenarios. It also refers to the unforeseen need of including interfaces for module interconnection.

Programmatic This final type of risk has more to do with the resource management and planning. A Designer has to identify the critical path for the development of the project. Contingency plans are needed to be able to execute according to planed without going over the budget should a unforeseeable situation happen.

An overview of the challenges of developing a mobile manipulator are presented in Section 3.1. The methodology used to propose a design that satisfies goals and limitations while marginalizing the challenges is outlined in (Section 3.2). Section 3.3 presents the Design Objectives defined for the developed system. Afterwards, hardware (Section 3.4) and software (Section3.5) design and implementation are presented. A discussion concludes the chapter in Section 3.6.

3.1 Limitations and Challenges

Hardware development is usually the most obviated part when developing a robotics platform to prove a concept for an algorithm. It usually happens when the focus is totally set on the algorithm. In some cases, there are prototype platforms put together only to provide proof of concept. These solutions are anything but robust because they lack a design methodology. When the focus is set on the development of the hardware, the problem of developing a platform grows and the software and higher level controllers are left for future works as the project matures. Although the chances of achieving more robust solutions increase, problems, such as portability of the solutions, arise. Solutions are particular and optimized to and for that specific hardware which in turn translates to having to develop adaptations or new implementations to port it to another platform.

Building a mobile manipulator, *i.e.,* a mobile robot with one or two manipulator arms, has been traditionally a major engineering project. The complexity of designing such a robot (system characteristics and capabilities) makes it a difficult task.

To gain more perspective, let us take the case of mobile robotics. It is common to buy a ready-out-of-the-box mobile base, equip it with some sensors (if not already included in the mobile base), and a laptop for control and sensor processing. This setting yields a fully functional system that can be used for several projects. This paradigm is also used in manipulation robotic settings.

Although it might seem straight forward to follow the same approach towards a mobile manipulation setting, there are many pitfalls. The complexity of such a system increases considerably when adding degrees of freedom and mobility to the system, let alone sensors.

Most of the challenges when designing a system are directly related to the available expertise, facilities and technology. In most cases, groups and labs do not possess a full range of all the needed resources to engineer a high-performance/high-end mobile manipulator. However, it is highly likely that there would be one specific area of expertise (planning, grasping, control, dynamics, etc.) in which the lab/group excels at, the lack of the other resources is what discourages them from conducting research on mobile manipulation.

In software engineering [59, 60], as well as in other engineering disciplines [41, 73], a common alternative to overcome the lack of expertise is to deal with a modular approach and using off-the-shelf components. The functionalities that are difficult or cumbersome to develop are acquired as black boxes and used only considering their inputs, outputs, and functions. The approach is not new and there have been some substantial work done

towards Off-The-Shelf component integration for robotics software [14, 22]. Until now, the approach has not fully reached the hardware nor the integral mechatronic areas of the field.

The modular off-the-shelf component integration approach has some limitations such as scarcity of candidate systems to integrate, complicated interfaces, and potential overheads in learning and/or modifying the module. Moreover, unlike software engineering methodologies such as XP, Agile, Scrum, etc., the development/design methodologies have significantly different approaches when there is a mechanical part to the project. Having mechanical parts involved makes the development stiffer and less flexible as the main outcome is not modifiable [1], thus the methodologies need to consider more time for planning and developing. For mechatronic design methodologies, it is paramount to consider all possible bottlenecks and issues before they arise and define a robust contingency plan to deal with them.

Among the advantages of using Off-The-Shelf components are time-saving, us of state of the art components without possessing the engineering expertise, and well defined performance at the outset of integration. Some drawbacks are that the subsystems might not combine in an optimal way, the detailed information and source code could be proprietary and non-accessible, the opportunities to develop new state of the art parts is missed, and potential difficulties exist in maintaining complex parts while lacking the detailed knowledge of their functioning.

3.2 How to Design a Mobile Manipulator

Most of the mobile manipulators currently used in research are custom-designed and -engineered. The major advantages of a custom approach are the overall control on the development process and the obtainable capabilities of the resulting system. The approach also provides better understanding of the system. However, custom engineering/design is expensive. Most of these custom systems take a long time to develop, need expertise on several fields, and cost a significant amount of money.

In several occasions that the lack of engineering facilities and access to proper tools plays an important role in taking the decision of building a mobile manipulator. A team taking a custom engineering approach needs access to specific and appropriate resources to design and build the robot. Therefore, a custom design is most of the times unfeasible for

[1] Unlike hardware, software is modifiable in a very agile way as it only needs a couple of hours to be changed without major costs of resources other than time. The main advantage of software over hardware is the fact that software is non-tangible and can be aggregated or clipped seamlessly

most research groups while an Off-The-Shelf approach would likely be a better option.

To plan, design, and build a mobile manipulator, it is necessary to consider the following sub-systems:

Mobile Base This might be, together with the manipulators, one of the most challenging modules to decide upon. Its main function is to move the whole system. The critical minimal payload has to be selected in order to achieve full mobility according to specifications. It is necessary to consider the weight of the whole robot and the maximum payload the robot is to be able to manipulate. It is also important to define the type of terrain that the mobile base is supposed to overcome and decide about the geometry of the robot. The mobile base usually defines how to mount the manipulator and therefore the reachable workspace and the dynamic stability of the robot while operating.

Manipulator(s) The most important features to consider for a manipulator are the payload, the weight of the manipulator and the integration of the power and control electronics. Ideally a manipulator should have a very high payload, low mass and no external components yet this setting is somewhat rare even now. While for small-sized robots no external components is quite conventional, for human scale and large manipulators this is still rare. It is also convenient to consider the control software openness, availability, compatibility, and the communication ports. It is standard to take into consideration industrial benchmarking characteristics: repeatability, tooling interface, dynamics, accuracy, and backdrivability. This latter is important for direct human interaction with the manipulator. When an arm is backdrivable, the joints do not lock even when powered down.

Computing The on-board computers of a robot often define its control autonomy. The variables to consider are its processing to consumption ratio, minimum processing needed, expandability, available ports and interfaces, software compatibility, memory, and available of custom BIOS settings. There are several options: from compact embedded computers (PC-104, SBCs) to full range compact motherboards (μ-ATX, mini-ATX, Blade Servers, etc.). If most of the processing power is needed on-board two specific constraints over the selection have to be considered: total power consumption and mechanical size. It is desirable to maximize the running time and make good use of the space in the hull of the robot.

Sensing The desired remote readings define the type of sensor that should be used. It is

crucial to set a specification of what needs to be measured early in the design stages as this will define the appropriate candidate sensors. Some general considerations are the sensor's "intelligence"[2], if it is active or passive, digital or analog, the interface, and the power requirement. Usually, cameras and proximity sensors are standard selections as they provide non-contact measurements.

Power Selecting the power source for a mobile manipulator can turn into an issue as the total amount of power consumption increases. Some of the things that come into play to select the power source are: how are different subsystems powered up, need of conditioning adapter, number of modules, and overall power demand (peak, idle, normal load.). There exist two options for selecting the power source: Either designing a general powering source or individually powering each subsystem. Moreover, the decision whether the system is to be tethered or use batteries. Having a unique power source has the advantage of equalizing the running time of the overall system and homogenizes the power interfaces over all the modules in the system. The downside is that some power conditioning interfaces will be needed for some modules.

User Interaction The user interaction is taken for granted or obviated depending on the applications of the system. Autonomous tasks do not require more than a mouse and a keyboard. On the other hand, teleoperation settings have to consider controller interfaces such as replicas of the remote robot, game pads, haptic devices, displays, wearable sensors, vision-based tracking, gesture recognition, etc. Moreover, interfaces should be selected from both an operator's and a developer's point of view. In most cases, a multi-modal input/output interaction should be considered.

Until now, only the main characteristics and features to keep in mind when selecting the different modules for the subsystems have been described. Then, how to integrate a system? This is a design question for which no exact answer can be given. However, a very effective approach is to follow the guidelines and always take into consideration the particular design requirements. The latter will define how to instantiate particular steps of the guidelines and produce a method to integrate the system. Procedure 1 shows the guidelines to produce a Mobile Manipulator.

The guidelines state that the Design Objectives should be defined first. The common practice in large projects is to define the goals and objectives. These define, in the most gen-

[2]An intelligent sensor is one which usually contains a microprocessor and can be configured and/ or programmed to give preprocessed data, modify the refresh rate, etc.

Procedure 1 Guidelines for designing a Mobile Manipulator

1: Define the Design Objectives
2: Review the Design Objectives and modify if necessary
3: Set critical minimum accepted values for principal variables
4: Define the system's architecture
5: Define Candidate lists for all subsystems
6: Select *Most Critical* Device (Usually Manipulator) and elaborate a tree of options
7: Evaluate the leafs' costs considering all limited resources
8: Prune the tree accordingly to remove non-viable options and select a candidate. If no candidate, redefine Design Objectives and follow this procedure again

eral sense, the steps to follow. Stating a clear and complete specification can pay-off later in the development process: less problems are likely to arise and the focus will remain on achieving the final outcome. After defining the Design Objectives, a through review should follow to detect possible pitfalls and conflicting objectives. Moreover, some problems are likely to arise if the design objectives are too lax or too tight: there will be too many or none implementation options, respectively. Reviewing the design objectives will yield consistent definition of the minimum requirements and will help elaborate the necessary contingency plans.

As the specification becomes solid, the designer has to decide on the critical minimum accepted values for the design variables. These variables are those which are set with tight requirements and tolerances to define the capabilities of the final product. Examples of design variables could be: mobile base payload, reachable workspace, and arm payload.

After the design parameters are set, the system's architecture should be decided. Definition of all the subsystems, inputs, outputs, and their functionalities have to be stated. At this stage, it is recommended to implement a design methodology to produce a well-defined architecture addressing all the design objectives and requirements. The most convenient methodologies to consider are Theory of Inventive Problem Solving (TIPS or TRIZ), Quality Function Deployment (QFD), and Failure Mode Effect and Critical Analysis (FMECA).

Once the architecture is defined, candidate off-the-shelf components have to be found for each subsystem. A complete (sometimes exhaustive) list of potential candidate components should be integrated. At this point, some specifications might have to be relaxed if no candidate component is found. For the most important modules, some selection criteria or methodology can be defined. Procedure 1 shows a method to select the mobile base for a human sized mobile manipulator.

Now that the architecture, functions, and candidate components are defined, the next

Procedure 2 Selecting a mobile base

1: Define $Factor_{Safety}$ to account for non-observable weights such as bolts, screws, harnesses, dynamic capabilities, etc.
2: Define $Load_{Operational}$ to include the weights of major subsystems such as Sensors, Computing, Manipulator and Power Source
3: Select a compliant mobile base with a payload of:
$Payload_{MobileBase} \geq Factor_{Safety} \times (Payload_{Nominal} + Load_{Operational})$
$\{Factor_{Safety}$ should be set according to the uncertainty of the extra weight. From empirical observation, it is a good practice to set $1.5 \leq Factor_{Safety} < 2.0$. Setting this parameter too low can yield a non-moving mobile manipulator; setting it too high would result in underusing of the mobile base's payload. Mobile bases with a higher $Payload_{MobileBase}$ are usually more expensive.$\}$

step is to organize and evaluate the alternatives for the subsystems. A straight forward approach is to use a tree structure. Each path is a particular configuration. The root (or roots if more than one candidate exists) is the most important critical component[3]. The next level gets as many branches as there are options for the second most important critical component and so on. Usually, some subsystems might be comprised of several components, in this case it is recommended to build a tree for the particular subsystem considering it as a system in itself and consider each version/option, as an alternative in the major tree.

Each path from the root to a leaf is an implementation alternative. At this point in the process, the critical variables from the specification have to be brought into consideration to detect those unfeasible alternatives. The discrimination process should be done looking out for general problems with completion times, budget, availability of resources, availability of tools, tooling requirement, interface design, etc.

The designer can now decide which is the best implementation for the project. Although there is never an optimal course of action, some options will be better than others depending on how they are evaluated.

3.3 Design Objectives and Requirements

Human interaction and the ability to solve human-scale tasks requires nimble human-scale arms suitable for contact manipulation. Moreover most of the interesting tasks require the robot to be mobile as the usual range of a manipulator is not enough for completing a mobile manipulation task.

The majority of mobile manipulators for human scale tasks are custom built and en-

[3]Usually those components with a particular functionality or those which are the most tightly defined in the specification

Requirement	**Value**
Data Connections	Wired and Wireless
Power Connections	Wired and Wireless
Power Source	Tether and On-board Power Source
Up-time	3 Hours Full Load (1 Hour Periods), 2 Hours Idle or Partial-Load
System Powering	Central power source (equalizing running time for sub-systems)
Weight of Manipulators	$\leq 60kg$
Manipulators Features	Nimble, Human-like Workspace, No external components, agile manipulation, backdrivable
Computing Power	Running resource intensive algorithms, inter-process communication
Computing Features	Expandable, Hardware for Video capture/interface, small geometric footprint and form-factor
Sensors	at least 2 cameras and 4 mounts, 2 on arms and 2 on body
System latency	network introduced delay $\leq 10ms$ and between processing nodes less than $\leq 50ms$
User/Control Interaction	Teleoperation Haptics, Debugging Joystick, keyboard, dual haptic feedback 3D device.

Table 3.1: Table of minimum requirements for the Mobile Manipulation System

gineered towards specific applications. The procedures followed to develop those systems are expensive in terms of time, money, engineering resources, and knowledge. Moreover, they fail to provide a generic platform for research as they are only available to a few (the developers) and replicating them is practically impossible.

In this case, the main design objective is to develop a mobile manipulation platform capable of performing human-scale tasks. The system should be built without the need of particular engineering facilities. The robot should be suitable to use for different applications such as algorithm development for visual servoing, teleoperation, human scale interaction, and parallel processing. In the general sense, the platform will be used for task prototyping with a particular focus on space robotics. The system is intended to be uses for diverse applications spanning a wide range of research areas. Therefore, the developed system should be flexible, modular, expandable, and robust to reconfiguration.

Because the Design Objectives of the system are broad, the requirements cannot be defined tightly. Instead, there is a minimum boundary for each requirement. The objective was to design an untethered system that could run for at least 5 continuous hours under 60% (normal) load. Preferably, the system should run on a central power source to equalize the running time. It is also desirable to have the option to run tethered from a wall plug.

The arms weight should be around $60kg$ in total. The manipulators should be capable of nimble and agile manipulation in a workspace similar to that of a human. The overall robot requires to be nimble, lightweight [4] and comparable in size to a small person. Computer vision algorithms have to be run on-board together with the motor-control, data fusion, and safety daemons. A connection for monitoring or doing remote computations should be available. These requirements are summarized in table 3.1

We define a nimble and agile manipulator as one featuring small moving masses (low inertia) an capable of high accelerations. Usually industrial grade manipulators have heavy links and embedded gearboxes and motors which yields an overall heavy and high-inertia configuration. Low-inertial arms do not require large motors which in turn reduces the overall weight of the arm.

One other parameter that is important to set in teleoperated systems is its latency. This characteristic is sometimes not considered in the system's description as is not crucial to include it. The latency of the system is the critical time between a command is issued and then executed. For teleoperation, latency becomes a very important factor; it is desirable that the system responds as fast as possible (low latency). In a teleoperation system with Low latency, communication delay can be studied independently.

3.4 Hardware Design

The robot was thought to have a configuration compatible with a variety of human scale tasks. This objective was translated into some design requirements of geometry and hardware integration. The selected configuration should allow to perform several tasks, simulating a person sitting on a rolling chair. The setting provides a reachable space suitable for picking up objects from the floor and for manipulating objects over a table-top. The system was designed to be reconfigurable in the number of manipulators: one or two. Either configuration considers putting the arm in such a way that the workspace reaches heights between 25 and $175cm$. The workspace of the two-arm configuration should resemble that of a human's.

3.4.1 Rationale

The list of candidate hardware submodules was integrated and evaluated. Table 3.2 presents all the candidate components for each subsystem. In some cases, there are sub-assemblies

[4] A lightweight system is that which is comparable to a large person's weight, say around $200kg$

Subsystem	Candidates
Manipulators	Barrett WAM®
Mobile Base	Segway RMP 200, Segway RMP 100, Segway RMP 400
Computing	Laptop Computers, Server Blades, Headless μATX Mother Boards
Sensing	Multiple models from PointGrey, Webcams
Power	Batteries (Lead-Acid, NiCd, MiMH, Li-ION), Wall connected tether.
User Interface	Game Pads, Joysticks, Mouse, Keyboard, Haptic Feedback Controllers (Omni Phantom, Omni 6DOF).

Table 3.2: Table of the candidates for each subsystem of the Mobile Manipulation System

which were pre-designed. The explanation for these cases is omitted (*e.g.,* computer hardware).

Although there are various options of manipulators and mobile platforms available commercially, only a few seem suitable to combine. Robot arms are usually thought for industrial environments; they might be bulky and feature a heavy external amplifier cabinet. The later should be avoided when possible because having modules with external components reduces the available space and payload on-board. Moreover, most arms have proprietary control software which cannot be modified. On the other end, mobile bases usually have payload capacities which can barely accommodate an arm. Two Barrett WAM arms atop of a Segway RMP mobile platform were selected as the major modules to integrate. Both options are attractive because of their own characteristics.

3.4.2 Manipulators

The Barrett WAM (Whole Arm Manipulator) is attractive due to its low total and moving mass, backdrivable joints, and high efficiency cable transmission (low friction). Some drawbacks are the constant need of cable maintenance and the occasional need of connecting an external control computer. Other appealing features of this manipulator are the open interface and source code availability[5]. Unlike conventional arms, the WAM is designed to exert contact forces not only with the end-effector, but also with any link surface. This is made possible by the efficient, low-loss transmission enabling the use of the motor currents as sensors for torque. Using the torque it is possible to calculate contact forces without the need of putting force sensors all over the manipulator's surfaces.

[5]provided by the manufacturer

3.4.3 Integrating The Mobile Base and the Arms

The Segway RMP mobile platforms are appealing in general because of their significant payload-weight ratio, moderate dimensions, and market proven design. In particular, the RMP 200 was chosen because of its dimensions and payload. Its geometry allows to place the arms directly on top. The resulting reachable workspace satisfies the specification. One issue with the platform is that the manufacturer does not provide open source controlling software. However, it is possible to design and run a personalized controller. The RMP was adapted a caster wheel on the front and a couple of safety back-supports because the dynamic balancing feature was not used. The caster wheel would provide a solid third point of contact with the floor stabilizing the robot even when manipulating objects with the arms.

For the two arm configuration a custom base was designed. The assembly should maintain an overlapping workspace for two-arms. This overlapping configuration provides good dexterity to manipulate objects in the front of the robot. Some kinematic simulations were done considering the volume of the workspace of each arm. Optimizing such overlap and dexterity of a two-arm configuration is a research topic in itself. An empirical human-like configuration was selected as it satisfies two main constraints: simple mechanical design and resemblance to the morphology of a human torso [43].

Arms are positioned in a 90° angle from one another, with both base-plates perpendicular to the ground tilted sideways. This is better illustrated in Figure 3.1. By empirical observation, it was assumed that the overlap achieved in this configuration was acceptable for two-arm manipulation in front of the robot. The configuration resembles the human torso while keeping the design of a base simple. In fact, the two arm mount was built from plywood without the use of specialized tooling other than common household carpentry tools. It is recommended that future iterations of the platform evaluate alternative configurations for the arms, *e.g.,* varying the angle between base plates. For this purpose, it is necessary to design a more rigid and lighter mount that allows fast reconfiguration.

The resulting geometry is compact. The space between the Segway's wheels, below the top mounting plate is available to encase computers or sensors. The space behind the base plates of the the arms is also available. Although compact, a minor drawback of the two-arm configuration is that the robot is still a couple of centimeters wider than a single-sheet door frame. The arm-mount should be redesigned if the robot is to be used in common human environments with single width doorways.

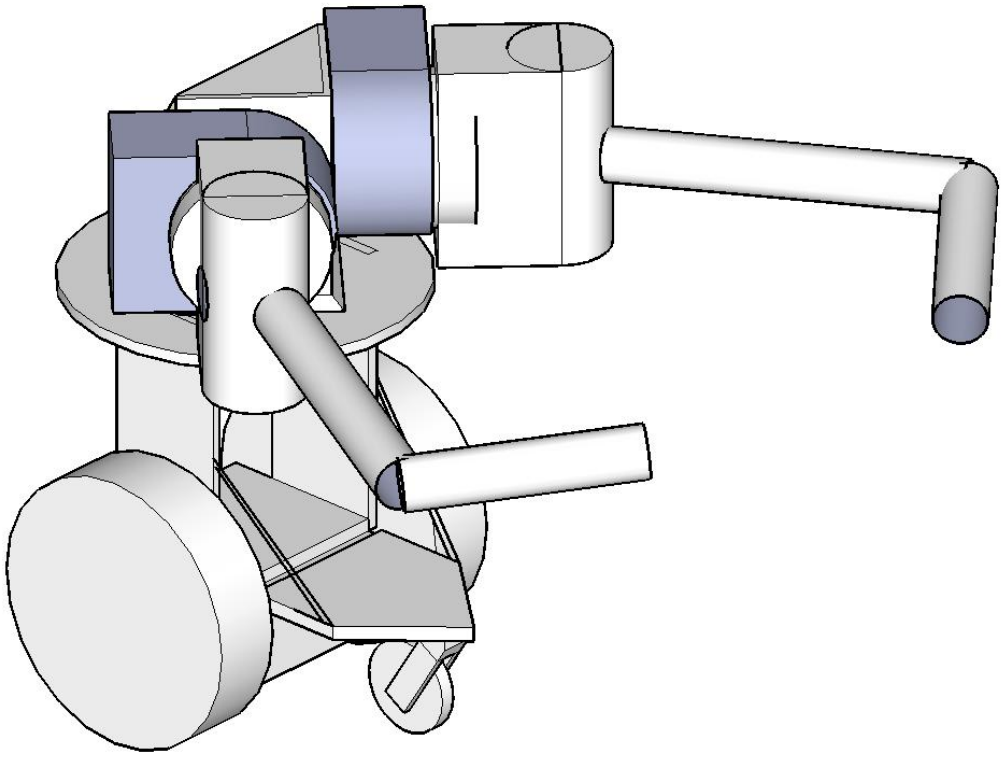


Figure 3.1: CAD model of the two arm configuration of the integrated system

3.4.4 Computing

Computing hardware found in lab robots ranges from specialized embedded solutions to standard top-of-the-line laptops and servers. Usually, computing hardware is directly related to the processing power and mobility needed on-board the robot. Most of the times on-board computing is needed for filtering, preprocessing, low-level control, communications, and resource management but this can also be done with a tethered system and an off-board computer. It is important to evaluate thoroughly the need of on-board computation as it demands a trade-off between the robot's running time, on-board available space, weight, and its mobility.

A popular choice for on-board computing are single board computers (SBC), in particular, the PC-104 series. They feature a compact size and various expandability options. These SBCs have the advantage of having x86 architecture[6] in a very small board. A downside of using embedded computers is that the most powerful recent processors are not available. In fact, PC-104 SBCs are usually a couple of generations behind consumer

[6]Lately, some other architectures such as ARM, have been gaining popularity

computers.

Another parameter to considering is the bandwidth of the processed data. Sensors for touch, proximity, localization, etc., can be be very well managed with an embedded computer. On the other hand, sensor arrays, *e.g.,* cameras or depth-map, provide a large amount of information, thus the bandwidth to read and pre-process increases significantly. It is then necessary to use the latest microprocessors, and in some cases, GPUs. Laptop computers can be used in some cases when only a few high-bandwidth sensors are used. If the system is using more than a couple of cameras, it is necessary to consider a custom made workstation. It is possible to find inexpensive workstation motherboards with small-form-factors like mini-ATX or μATX.

The designer has to consider the possibility of expanding or updating the system by adding extension boards[7]. Together with the hardware expansions, considerations have to be taken on the overall compatibility with other software and other existing modules. Laptops are a good solution in case of a mobile medium sized robot, although expanding them might be more expensive than a regular desktop workstations.

For the system, customized workstations were chosen for a variety of reasons. Most of commercially available motherboards allow personalization on very low levels to optimize the performance of the computer. Setting and managing the voltages and frequencies on the motherboard allows to minimize the power consumption while maximizing processing power. It is also possible to mount different compatible processors and peripherals into the board, thus keeping the system modular and open for hardware updates. Although laptops have become as powerful as desktop workstations, the main advantage of the latter is its modularity and expandability.

Expansion capabilities worth considering are network interface cards, port expansion cards, data acquisition modules, and the available ports on the mother board. Most commercial active sensors and actuators feature network, USB, and/or Fire Wire connections. Their In-house-developed, field, industrial, and passive counterparts usually need some signal conditioning.

A downside of using desktop workstation computers in a mobile robot is that the user interaction peripherals usually have to be removed (monitors, mouse, and keyboard). This makes it difficult to debug and monitor the system. Laptops do not suffer from this problem as they are fully integrated. Setting up and configuring headless computers is somewhat

[7]Different interfaces are available like PCI-E, PCI, USB, Express-Card or PC104, PC104+ expansion modules

cumbersome, but once they are running they can be accessed seamlessly from another node connected through a network interface.

Currently, the robot features only one of the two quad-core computers planned. This computer is used for general resource management and processing (video and sensory input, motion planning and teleoperation). This computer also connects the two embedded computers, each controlling one robot arm. In the future, the robot will incorporate another computer which will be used for vision processing leaving the current computer to deal with motion planning, communications, and system management, effectively separating the processing load on-board the robot.

3.4.5 Sensors

The robot currently uses four cameras when using two arms and three when using only one arm. These cameras feature high speed Fire Wire (IEEE-1394b) and can work in various modes. It is possible to use them for fast image acquisition (up to 200Hz on VGA).

In the future, the robot is planned to include a pan-tilt unit (Biclops by TRACLabs) to position a human inspired camera pair. These cameras will be uses for navigation and for eye-to-hand visual servoing. One camera is mounted on each arm. These are used in eye-in-hand visual servoing. These four cameras (three in the case of the one arm setting) are the major sensory subsystems of the robot[8]. Including other sensors, such as touch and force sensors, is left for future improvement of the mobile manipulator.

3.4.6 Power

A mobile manipulator has multiple subsystems which need to be powered. For less complex systems, buying battery-powered subsystems (*e.g.,* laptops, sensors, mobile base, etc.) is an acceptable option. For larger systems, such as this, a general power source feeding all the subsystems is a sounder alternative. Having a unique power source equalizes the running time for the whole robot while minimizing the number of different batteries that have to be charged.

The system has a universal AC/DC power supply units (PSU). Such PSUs were selected because they can work from both AC or DC[9]. In turn, it is possible to power the whole

[8]The arms' encoders, current sensing, or the Segway's IMU and odometer are not considered as they are part of other subsystems

[9]These PSU feature a rectification phase upfront the voltage converter. These power source units are very attractive as they can work both from AC and DC (provided that the line voltage is above the switching threshold. As these PSUs are not on purposed designed to work from DC, it was necessary to test for warming and/or loading on the rectification bridge to assure that the integrity of the unit is not compromised and working within

Feature	Required	Achieved
Idle Running Time	5 Hours	9 Hours
Loaded Running Time	3 Hours	≈4.5 Hours
Nominal Voltage DC	≥100 V	144 V
Max. Discharge Current	11 A	25 A
Max. Peak Load	1.5 kW	3.6 kW
Weight	≤ 30 kg	24 kg

Table 3.3: Table comparing Required and Achieved Power Source Features

system from a power bus carrying AC (from a tether to a wall electricity outlet) or DC (from an on-board bank of rechargeable batteries). The use of an AC tether permits the robot to function continuously without interruption. This is useful when debugging or running experiments in reduced controlled environments.

Different battery chemistries were evaluated. Capacity, energy density, depth of discharge, inner resistance, recharging cycles, and price were the considered variables. For the scope of this work, a "robust" battery is one that can tolerate overcharge, deep discharge, and mechanical shock. Lead Acid, NiCd, NiMH, and Li-ION battery chemistries were tested as each of them feature attractive characteristics. NiCd was the final selection because of its combination of high robustness and affordable price. Battery packs of 36V and 8Ah capacity were used.

Table 3.3 shows the summarized on-board power source unit required and achieved values. The total nominal capacity of the power suply is 1152 Wh and it has been tested to be able to run the system until deep discharge of the batteries.

One of the heaviest subsystems in the robot is the power-source. The batteries were placed as low as possible in the robot's hull to reduce the effects of having this large mass affecting the stability of the robot.

3.4.7 User Interaction

Teleoperation requires to provide different ways to issue commands to the robot. An interface better suited than a mouse and a keyboard is recommended but not necessary for this case. Different interfaces provide advantages for different purposes. From the user's point of view, it would be desirable to have an interface to issue commands with ease. A variety of input devices are available: from a common mouse and a keyboard to a force feedback enabled devices. The user interfaces can also be implemented using virtual controllers (*e.g.,* Gesture Recognition).

specification.

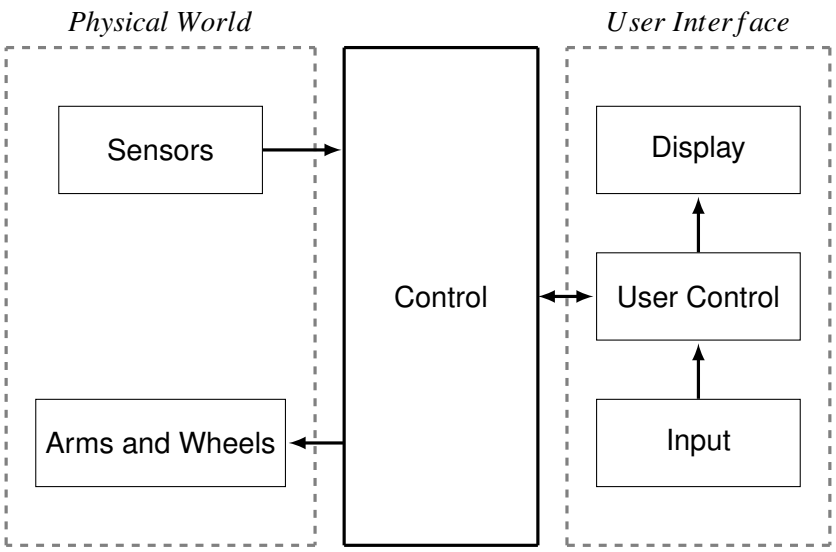


Figure 3.2: Diagram for a simple mobile manipulator with minimal setting.

In this case, the system can use multiple different user interaction devices. Haptic Force feedback for teleoperation is an axial components of this research project. A Sensable Omni Phantom® was selected as an interface for the mobile manipulation platform. It provides an acceptable force feedback settings with attractive portability features. In future iterations, other devices such as game pads or joysticks could also be used. These are suspected to provide faster "game-like" interfaces while keeping costs low. Such input devices are pr particular utility when debugging the mobile base or the arms.

3.4.8 Hardware Architecture

The mobile manipulators presented in the previous chapter have small hardware variations between each other and therefore a generic architecture can be identified. Most variations occur in the main computing/communications modules, available sensors, on board systems, and intermodule interfaces. For example, some robots have a general bus used to communicate all the submodules, while others have local connections organizing the sensors and actuators into semantic groups.

The usual architecture includes a central computer to which the arm(s), mobile base, sensors, and communications are connected. This configuration may lack robustness and modularity at times, but it is the most simple to implement.

Another aspect of the system's architecture is the geometric and spatial arrangement of the on-board components. A good strategy is to organizing the modules of the system based on their size or importance. For this system, the first two (or three) modules to consider are

the mobile base and the arms. Afterwards, the power source was set and the placement for sensors and their mountings was decided. Sensors are usually set up to maximize their sensing utility while keeping occlusions minimal. Finally, computers were placed in the more protected places of the robot. Once all modules are accommodated, it was possible to harness and connect the modules to one another.

An attractive feature of the Segway RMP 200 which was selected for this implementation, is the protected space between the wheels and below the top-mounting plate. This space is placed low and close to the axle of the robot serving to counterweight the arms on top while keeping the contents protected. Batteries, power supplies, and computers were placed in this space.

The computers were cased into small lightweight standard miniATX cases to protect the boards from being directly exposed to the environment. The four wired ports of a standard wireless router were used to connect the arm's embedded computers and the processing workstations into a local network. The wireless connection was used to connect off-board and monitor computers to the system.

Figures 3.2 and 3.3 depict the general architecture of a teleoperation system and the instantiation of the system implemented. Connections of all the subsystems and the general hardware are shown. Figure 3.4 shows actual pictures of the resulting implementation and the geometric arrangement in the two settings: one- and two-arm.

In the future, the robot can be equipped with other sensors and also expanded with a larger bank of batteries to extend the time it can work untethered or include more processing nodes if needed.

3.5 Software Design

In this section, the definition of the software framework is presented. The design objectives in this case center in producing controllers, module-, and user- interfaces for the teleoperation system. The controllers were defined as the pieces of software providing sensor preprocessing and correlation, actuator commanding, system stability, and management. Module interfaces are software modules which provide the messaging layer and provide higher level processing and information correlation. User interfaces are the software modules closed to the operator and provide the interfacing layer for command input and display.

In general robotic projects have custom made control software to cover specific functionalities. The software in most cases is monolithic and tightly integrated reducing its

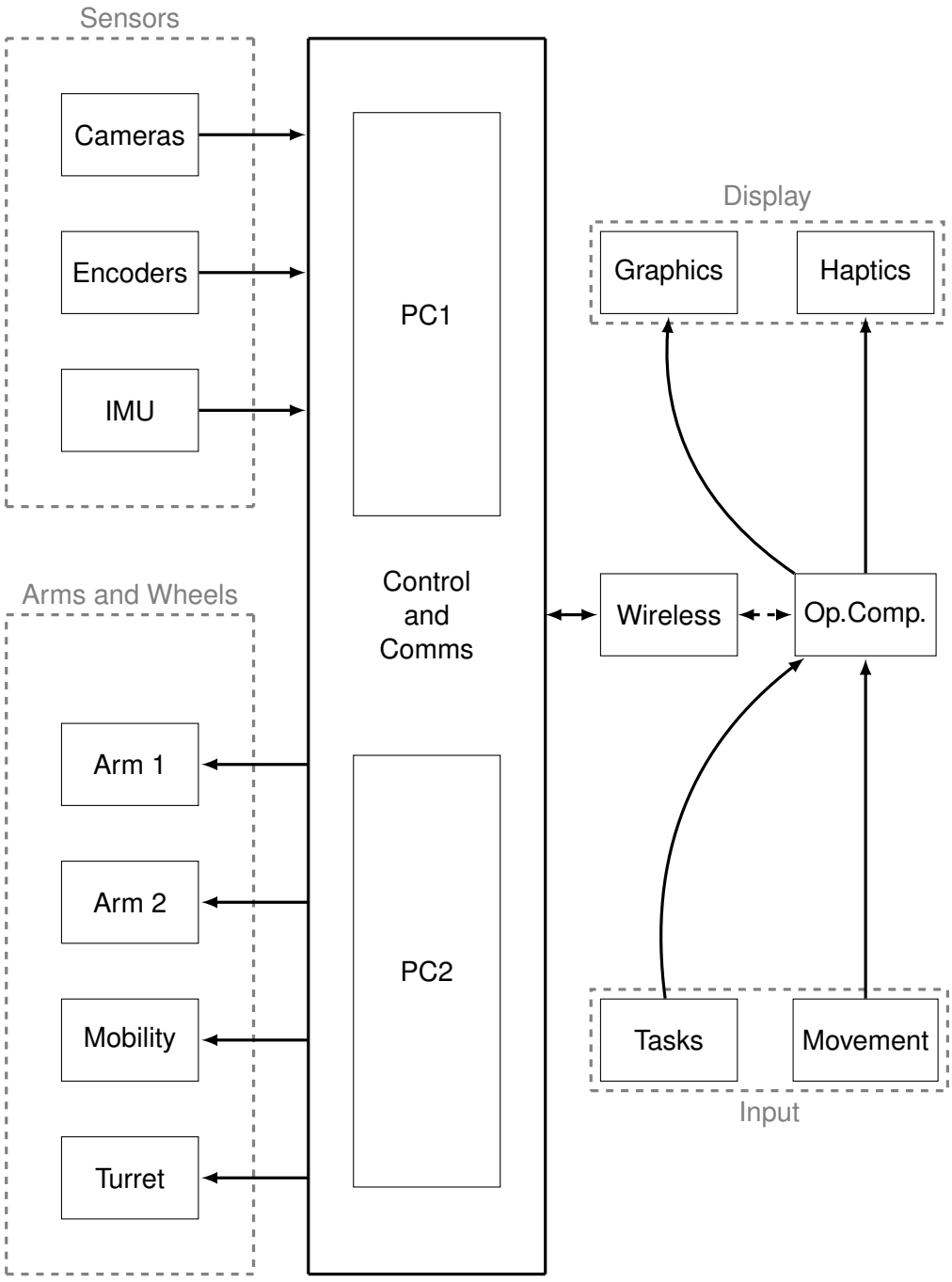
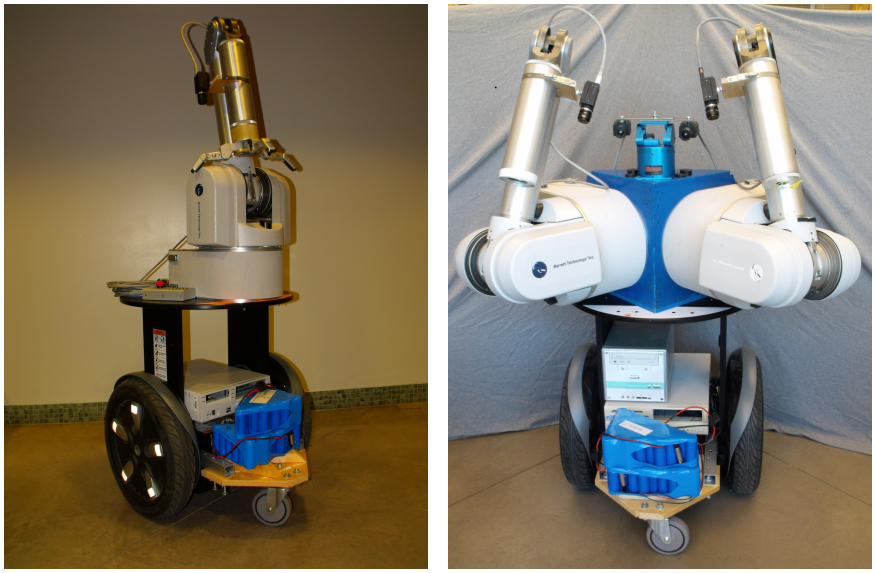


Figure 3.3: Diagram for the two-arm mobile manipulator setting with all the planned add-ons

portability and modularity. Functionality is privileged over having a long term consistent and expandable framework. Moreover, most robotics applications require time-critical response and solving particular implementation problems.

Robotics frameworks usually define what languages to use, which operative system to



(a) One Arm Mobile Manipulator (b) Two Arm Mobile Manipulator(Full System)

Figure 3.4: Pictures of the mobile manipulator: (a) Picture of the one-arm mobile manipulator setting with minimal setting and (b) Picture of the two-arm mobile manipulator setting with all the planned add-ons

install, and how to achieve inter-process message passing. These aspects can be set early in a design stage for the software. Later during the implementation phases, specially when adopting new technologies, the framework is likely to be expanded as new requirements will arise. Architecture, processing, critical paths of information, and robustness are aspects that are difficult to detail early and therefore have to be taken care of near the deployment phase.

It is common practice to use scripting languages, *e.g.,* Python, for prototyping, but compiled languages, like C/C++ are favored when deploying a final implementation. Scripted languages can still be used for higher level functionalities while lower-level controllers (running control loops or hardware message-relaying) have to be implemented in compiled languages.

Most operative systems used in robotics consider the use of real-time characteristics. Also, to keep things modular, using interprocess communication is required. Because of these two reasons, Unix-like operative systems are usually favored. There are some high-end alternative embedded operative systems such as VxWorks® which can also be used. The main advantage of a Unix-like OS is the ability to fine-tune and manage the computing resources, and easing the implementation of parallel and distributed processing.

Message passing and interprocess communication is an attractive tool which enables modularity in software. Having separate defined nodes keeps functionalities organized an

Component	Candidates
Languages	C, C++, Matlab, Octave, Python
Operative System	Linux (Fedora, Ubuntu, Slackware)
Message Passing	Message Passing Interface (MPI), Parallel Virtual Machine (PVM), Robot Operative System (ROS)

Table 3.4: Table of the candidates for the components of the software for the Mobile Manipulation System

canned. Modules can be spawned or killed during runtime providing robustness to the system. Faults can be troubleshooted without having to bring the whole system down.

3.5.1 Rationale

For the software that will be deployed in the teleoperated mobile manipulator, no specific methodology was chosen. The framework was defined to be extended with new modules providing particular functionalities.

The same approach as for the hardware was followed. A table of candidates for the operative system, message passing interfaces, and languages was integrated. Unlike hardware, the general architecture is not defined explicitly except for the parts dealing with hardware interfaces and drivers. The table defines the overall basics of the framework, and although it provides some constraints, it is by nature lax and should be tightened, extended, and defined in detail for particular projects. Table 3.4 shows the basic imposed constraints on the software development and the overall "playground" for the teleoperated mobile manipulator's software.

3.5.2 Building the Framework

There are a couple of approaches towards defining frameworks from the point of view of software engineering applied to robotics. These works focus on defining reusable code and ways to guarantee robustness in the applications. Bensalem [14] and Brugali [22, 23] focus on autonomy, safety, and robustness but fall short in runtime contingencies and delay between teleoperating nodes. Both try to bring common practices in software engineering (component based software engineering) to robotics while keeping interest in the control real-time requirements.

When selecting the languages to develop software, it is desirable to consider those that are compatible (*e.g.,* Matlab can be interfaced to C/C++ using mex). It is important to define at least one compiled language which allows low level control and one scripting language for both prototyping and implementing higher level functionalities.

An important factor in selecting the language is the available support. A language having a large user community of is more attractive than a language which promises to have interesting features but only known by few people. Having community support gives two immediate advantages: support while developing and larger audience can use, test, extended, or debug the implementation. A third advantage is the availability of third-party software.

Selecting an operative system to install in the processing nodes defines the default available functionalities for them. The most common alternatives are application specific embedded or Linux-based operative systems. Both alternative sometimes feature real-time capabilities. Linux offers several attractive features which are not common on other operative systems: capability to control the application at any level, message passing and multiprocessing libraries are optimized and native to the OS, and it is open-source.

Most message passing and interprocess communication libraries are meant for distributed and parallel processing. They implement the functionalities needed to communicate processes while leaving the overhead of defining the channels, protocols, handling, and implementation of higher order functionalities (*e.g.,* services, logging, etc.) to the user. As the number of message types increases and the architecture expands, it becomes more difficult to handle the information flowing between modules.

Normal message passing libraries like MPI and PVM do not provide ready to use communications. If using these libraries, it is necessary to define ways to deal with incomplete messages, blocking interfaces, logging, etc. Fortunately, there has been a strong initiative from Willow Garage [62, 63] to develop a robotics-specific framework/toolbox (ROS) which takes care of these implementations.

3.5.3 The Software Architecture

An issue with off the shelf components is that the selected modules may use different computational architectures. Moreover, the computation requirement for vision tasks, robotic control, teleoperation, and system management could be too big for a simple computer to handle alone. This requires the framework to detach the hardware abstraction layer from the control and resource management layers. Parallelizing through process intercommunication allows to run different system modules from different processing nodes while homogenizing the access to the hardware and processing resources.

In a first approach, Parallel Virtual Machine (PVM) [36] was used to distribute the processing load across a network of computers. PVM offers to bypass architectural and

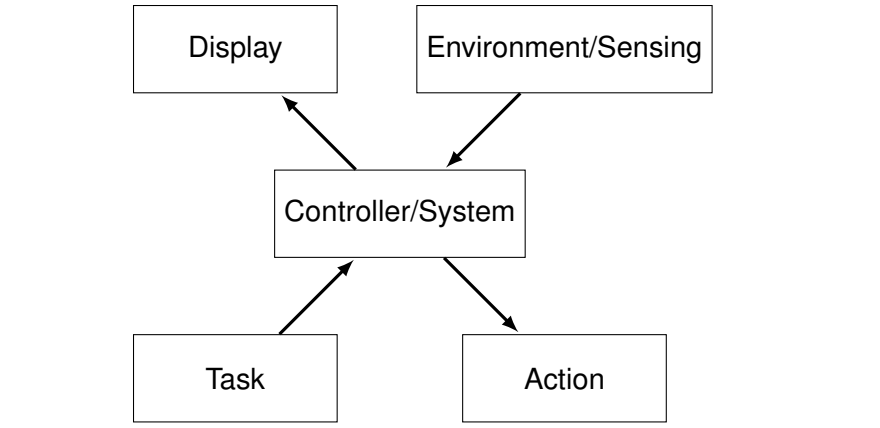
networking issues by setting-up the machines as one single computer. The library is intended for generic parallel processing thus requires some additional customizations in order to provide full functionality for the mobile manipulator. Time-stamping, bagging, and buffer inversion had to be custom implemented. Moreover, a message protocol had to be established between each pair of modules. Since pipelines are not supported, some issues arose for video streaming.

Because of the previously mentioned downsides found when using PVM, it was decided to use Willow Garage's Robot Operating System (ROS) for future developments instead of PVM. ROS provides hardware abstraction, message passing, process intercommunication, time-stamping, logging, custom message definition, multi-language support, debugging, and monitoring tools. ROS is robotics oriented and therefore provides these critical functionalities, unlike PVM. From the point of view of a robotics developer, ROS outperforms PVM. From the implementation point of view ROS has a couple of downsides: runtime execution and automatically-generated-code overhead. Code developed in ROS might be slightly slower at runtime because of the overhead introduced by the self compiling routines to provide the framework services [10]. Moreover, some computers were not suitable to run ROS (the computers controlling the arms) due to conflicts with Real Time aspect and insufficient processing resources.

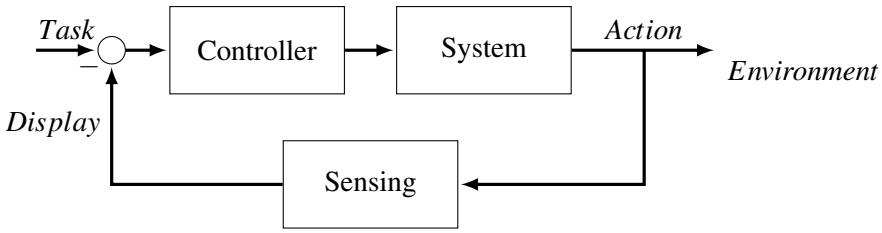
In figure 3.5 a generic architecture is presented. It depicts a control loop which takes into account inputs from the world and from a user (if any, *i.e.,* in teleoperation). Figure 3.6 shows a refined version of the previous diagram. This architecture representes a teleoperated mobile manipulator with modules for predictive display and visual servoing.

The main software integration issue to overcome was to effectively modularize the arms' control computers. These computers were set up with particular operative systems, custom hardware, and particular software difficult to updated regularly. These computers are very different from those used for general processing. Using PVM in these computers was as difficult as with the processing computers. Migrating to ROS brought some issues as it was not possible to install on these platforms. It was necessary to make a custom TCP/IP server for each arm. This server connects to an interface enabling each arm as a node within ROS. Future implementations assume that both arms and mobile base will be controlled from a single computer capable of running ROS and real time applications.

[10] ROS code, in the worst case, can have an execution time overhead up to $10ms$, while PVM does not add significant latency to the code



(a) Semantic



(b) Block Diagram

Figure 3.5: Basic Software architecture for the mobile manipulator. (a) shows the semantic point of view of the architecture. (b) depicts the same architecture using a classic control block diagram.

3.5.4 Notes

Some software tools were used during the development of the platform, and some others might become useful in future developments. Most of these tools are for computer vision, control, and simulation. Some of these are already integrated into ROS. Future projects should privilege the use of those tools that are already integrated into the framework. Some functionalities are not yet considered into ROS. In this case there are two options: either integrate them to the framework, or use them as stand-alone libraries. The former should be preferred to the latter. Although integrating libraries into ROS might be time consuming, the results are not only useful for the project but for the robotics community as well. A risk of using stand-alone libraries are the potential technical and operational issues, *e.g.,* incompatibilities.

Some of the toolboxes/libraries considered are:

XVision [38] A library for computing vision implementing basic functionalities. Provides a pipeline for video enabling multi-access to video feed and images.

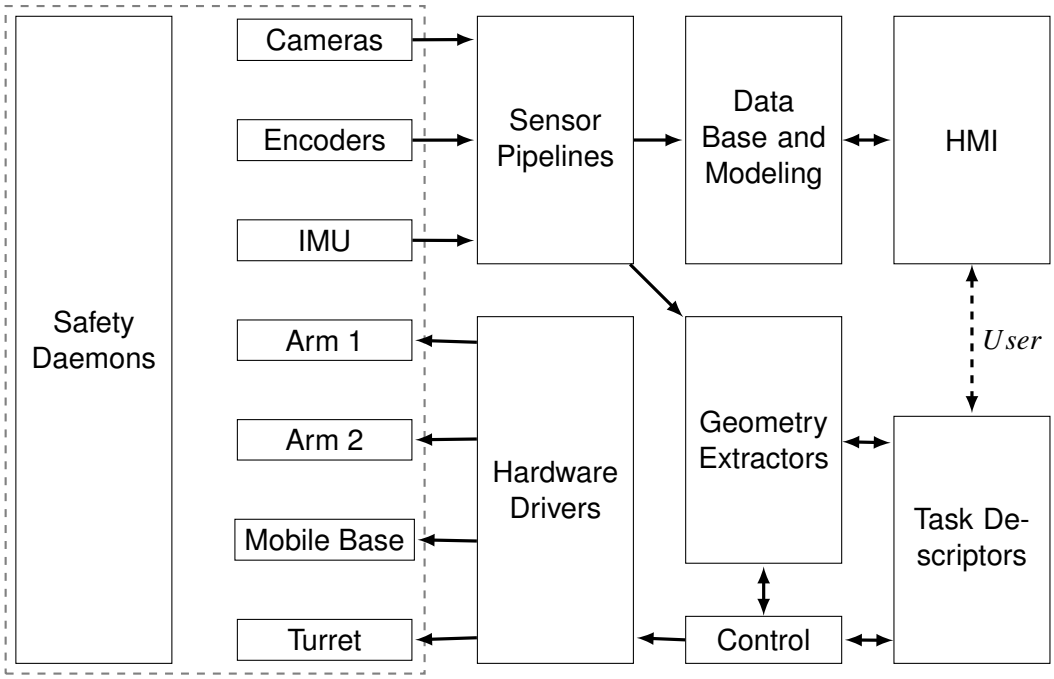


Figure 3.6: Software Architecture used for mobile manipulation with Predictive Display, Visual Servoing and Teleoperation

OpenCV [19] Currently the most widely used standard library for computer vision. It is implemented within ROS. It features several standard methods for image acquisition, image processing, filtering, etc.

ViSP [50] INRIA's visual servoing toolkit including common standard tools. These tools feature basic routines for arm control and computer vision.

OpenRAVE [32] An environment used for simulation purposes. Provides a way to safely test control algorithms and basic correctness of control and teleoperation code. It is integrated into ROS.

OpenMAN [45] A library developed for general arm manipulator control. Currently supports: WAM, CRS, IRB, FANUC manipulators.

The controllers of the robot are organized in different hierarchical layers. The layers closest to the hardware run the innermost control loops. The controllers for high level processes, such as vision and display, run in the outermost control loops and on top of the inner layers. Those controllers closest to hardware run faster than those on a higher abstraction layer. For example, the WAM controllers run between 500Hz and 1kHz, while the visual servoing loops run maximum at 60Hz.

In the future, safety features have to be implemented into the robot. Safety daemons should be implemented as low-level controllers parallel to the normal and user controllers. These daemons monitor the conditions of operation, troubleshoot module faults, and assess the malfunctions. At the present stage all actuation subsystems have minimal safety monitors limiting operation and emergency shutdowns.

3.6 Discussion

Designing and building a mobile manipulation system for teleoperation is a complex task that used to be reserved for those with access to engineering facilities and technical knowledge spanning several disciplines. The availability of lightweight fully integrated "embedded" robot arms and high-capacity payload mobile bases make it possible to integrate a mobile manipulator using these modules. It is possible to build a system from Off-The-Shelf components, with minimal ad hoc. design having functionality comparable to a custom made system.

This chapter explored the Off-The-Shelf Component Integration methodology to build a mobile manipulation system. The process is less intensive than design from scratch but demands certain ability to identify programmatic, technological, and operational challenges. Key features to pursue in the design of such system are modularity, flexibility, and expandability. These three characteristics make a system general enough for different applications and guarantees to provide a platform for teleoperation and mobile manipulation research.

Hardware and software design are both largely benefited from Off- The- Shelf component integration. This integration methodology makes the design process more agile and lets designers focus on the hardware interactions and selecting the appropriate modules the different subsystems of the mobile manipulator. Mobile base, manipulators, computing, sensing, power, user interfaces and software tools need to be selected taking into consideration their compatibility.

Application specific projects have tight and fixed objectives which do not leave a lot of room for expansions. This makes these systems confined to a few applications. Relaxing the requirements and having generic objectives leave a project unconstrained with various possible applications. It is important to set solid objectives and define a framework that guides the development of the project.

Software design is in itself a field of computing science, covered by Software Engineering. For robotics, software engineering alone is not sufficient to encompass the knowledge

needed to develop useful software for a robot. Knowledge on hardware interaction, real-time applications, embedded software, and hardware architectures is primordial. All this helps select an appropriate operative system, define a software framework, use suitable languages, and provide a way to communicate different processes considering the hardware limitations of a robot.

There are multiple things that need consideration when developing a teleoperated mobile manipulation test bed and regardless of the advantages of Off-The-Shelf component integration, there are several downsides to the methodology, *e.g.,* proneness to failures due to design unawareness. Yet, methodologies like Off-The-Shelf component integration, make mobile manipulation more accessible to a larger audience, hence more research can conducted on this field.

Chapter 4

The Operator-Robot Interface

The previous chapter reviewed the process used to develop the test bed for teleoperation. Half of the solution is proposing an architecture and a methodology of design specific to mobile manipulation. The other half is what this chapter reviews: the controllers and programming ideas used to get the teleoperation setting to work.

Teleoperation of mobile manipulators is a viable control strategy in mission-critical applications such as hazardous material handling and space robotics. Most space related mobile manipulation studies assume semi-automatic control while performing actions in the remote environment and do not consider tele-operation. This lack of consideration is due to the potentially large communication delays inherent to inter-planetary settings and the way the delay affects the operator decimating his/her effectiveness to perform a task.

Some of the main solutions to the time delay problem is making the remote system autonomous, introducing control techniques such as the four channel architecture, and making the system follow a series of commands by introducing virtual reconstructions of the remote site with technologies like Predictive Display [47]. In the first case, the setting relies on the slave to be robust and be able to recover from failures, as well as to deal with a great amount of uncertainty (not to mention the need of high computation power on the remote site, either on-board or near the slave system). The second option cannot deal with large delays as the transparency of the system is decreased as the delay increases. This makes the task completion too difficult on the operator. Predictive Display takes care of the delay by making a local model for the operator in which the tasks can be performed, yet again, one has to take care of the delay by introducing some autonomy and robustness in the slave side because the virtual model can only be updated after new information, subject to delay, has arrived.

Although delay is an interesting problem, it is being addressed by various researchers.

Time delay should be tackled in later stages [1] of the development of the teleoperation system; first, it is necessary to ensure that the system can work and is suitable for operating without any delays. Then the main concern is to first provide the operator interface for the teleoperation system.

Teleoperating a mobile manipulator can be onerous on the operator as the number of degrees of fredom (DOF) to control is large. Let us take for example a simple 4DOF manipulator atop of non-holonomic mobile base. This robot features 7 DOF but only six can be directly controlled. Moreover, the system is likely redundant in some of its degrees of freedom. The inherent complexity of commanding the system demands to develop methods and interfaces for natural teleoperation of mobile manipulators.

This chapter outlines three different ways to command a one-arm mobile manipulator (4DOF atop of a non-holonomic mobile base) from a six-degrees-of-freedom input device capable of haptic feedback. This interface has different kinematics and dynamics from the arm, which makes the system asymmetric. This makes it necessary to solve the issues of master-slave motion coordination, discussed in section 4.1. Later, the schemes under study are presented in Section 4.2. Immediately following, in Section 4.4, manipulation and haptic helpers are introduced as a means of automating and relieving some of the burden of manipulating remotely. Finally, a discussion is introduced to conclude in section 4.5

4.1 Master-Slave Motion Coordination

When enabling the motions of the master device to command the slave, some issues arise and have to be solved for all asymmetric teleoperation systems. Unlike with symmetric systems, a simple as a 1:1 joint to joint correspondence cannot be directly implemented and at least a scaling factor should be implemented. In most cases, in addition to the scaling factor (mapping), a way to switch the reference point (anchor) should be implemented to map the full workspace of the slave to the available space on the master.

Figure 4.1 depicts how a master-slave motion coordination scheme works. The scheme should define how the motions of the master translate to motions of the slave. It should also provide a way to map the full workspace of the slave by using scaled inputs from the master. This inputs should be used to move the anchor and to issue local motions.

Mapping This issue refers to the way in which the motions in the master side are translated to motions in the slave device and viceversa. Either when using position-to-position

[1] Although latencies are to be introduced later in the development, the design considers this research area and controllers are customizable and modifiable

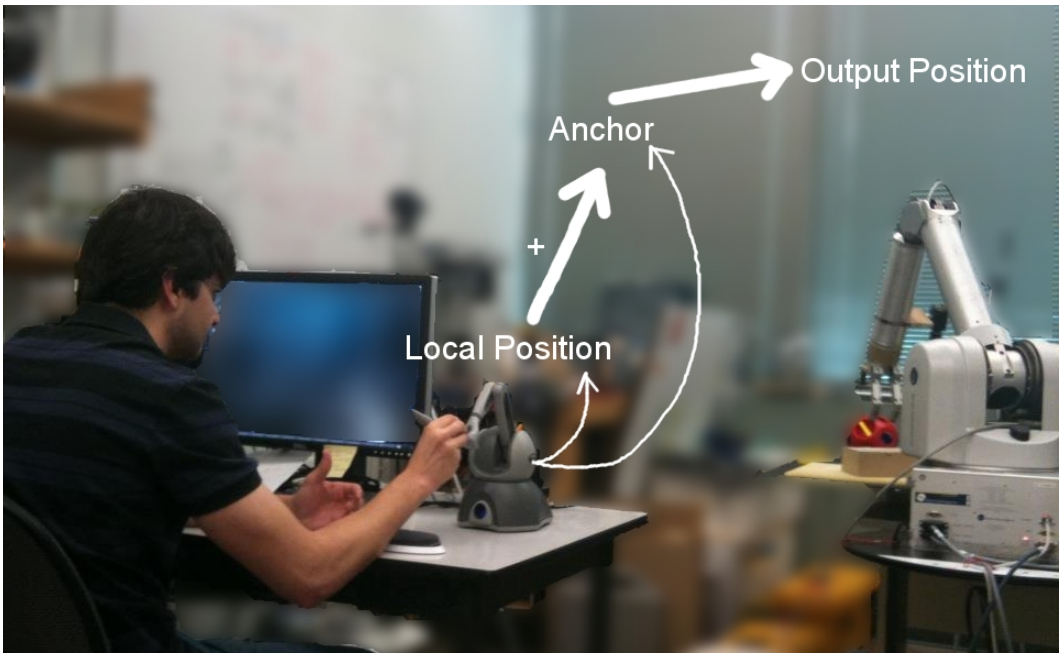


Figure 4.1: A Master-Slave motion coordination scheme should provide a mapping and a way to shift the anchor. The anchor and the current position of the master together with the mapping coefficients should ensure a valid output position is produced for the slave to move to.

(haptic devices like the Phantom Omni) or rate-to-position (gamepads or joysticks) devices there are three approaches, each with its own advantages and disadvantages:

Joint space to Joint space This approach maps direct changes in the values of joint motions of the master to changes in the values of the slave's joints. In most cases the motions in master and slave will not correspond due to the mapping. A Cartesian motion up may not yield the same displacement in the slave side. This issue becomes more evident when dealing with force-feedback as the resulting forces or torques are non-intuitive for the operator. The simplicity of this approach is what makes this alternative attractive. There is no processing needed or inverse kinematics to be solved as in the Cartesian mapping. Another advantage is that the operator can fully control the motions of the slave since the mapping enables the direct command over the configuration on the joint space. This approach gives up intuitive motions in favor of compliant and faster solutions on the slave. Feedback in this case can be implemented by rendering the slave's sensed torques and directly replaying those torques in the master's corresponding joints.

Cartesian to Cartesian Although the previous approach seems attractive, it lacks a the very important aspect of being intuitive. Having an intuitive mapping is desirable as it translates in less task load and training overhead for the operator. Using a Cartesian to Cartesian mapping guarantees that a motion in the master side will be translated to a very similar and corresponding motion on the slave. Having this feature comes at the cost of having to solve the forward kinematics for the master device and the inverse kinematics for the slave. The latter, introduces the need of taking care of deciding which solution to use (if multiple are found) and also dealing with singular configurations. While gaining intuitive motion command, the system has to deal effectively with the issues arisen by including inverse kinematics into the mix. Preferably, feedback should be implemented using the sensed forces on the slave and rendering forces on the master. In the case of force-feedback enabled systems, it is also necessary to calculate the forces that need to be rendered on the master using the torques and the Jacobian or forces registered by sensors on the slave.

Joint space to Cartesian This approach is mostly used for gampads or joysticks because is mostly useful when the motions are not commanded by a kinematic chain. In this case a joint value is mapped to a Cartesian axis or rotation. For force or haptic feedback the feedback on each axis is mapped to the matching force received from the slave. The approach is intuitive as far as the operator learns the mapping. When using gamepads, the feedback should be mapped from the slave's sensed forces to torques in the master device. Most of the times, this mapping is used with gamepads featuring vibration haptic feedback.

Cartesian to Joint space This kind of mapping would result in unnatural mappings and therefore is not used.

Anchoring Another issue is how to enable the user to reach the full workspace of the slave with the available workspace of the master. The two most intuitive options are:

Disengaging and warping the anchor As it is implied, in this case the motions of the master are used to move the slave and also to move the reference point or anchor. This anchor is used as the reference to calculate the global position to command the slave. At one given time, while keeping the anchor static, the motions of the master are used to render local displacements on the slave. If the reference point needs to be moved, then the direct control over the slave is

disengaged, and the motions of the master are used to warp (move) the reference point to a new global position. While disengaged, the anchor is updated to a new global position. Once the warping has finished, then the master's motions are again engaged to the slave. The **clutching** scheme — Presented in the following section — corresponds to this class. In the case when the master device has a bigger workspace than the slave an overall reduction factor can be used and therefore scaling up the slave's workspace. In micro-manipulation it is possible to define an overall scaling $1mm$ displacement in the master to $1\mu m$. More elaborate schemes such as using scaling together with the current position (**zooming**) are also possible but more difficult to implement[2].

Commanding to move the anchor with a rate In this case, the anchor is moved by issuing rate commands. This rate can be rendered using the master device while mapping its motions to rate, or issued by another device such as a keyboard. In the latter, a key-press would add some offset to the anchor. Using schemes that incorporate rates makes it possible to keep the master and the slave engaged at all times. Using a game-pad or joystick based schemes fall within this category. In the following section, two schemes under this class — Differential-End-Zone and Position/Rate switch — are explained.

In order not to provoke the reference point to become lost, it is necessary to limit or filter the anchor's valid values, *e.g.,* enforce the anchor to always rest within the slave's reachable workspace. If a command from the user causes the anchor to go outside of the valid space, then a recover rule should be used to guarantee compliance at all times.

4.2 Intuitively Commanding the Robot

As visual feedback has become an essential part in teleoperation in remote environments and visual sensors become inexpensive and more accurate, it is natural to include several of these in teleoperation settings. Although giving rich information, visual feedback lacks most of the kinesthetic information humans usually use as cues when manipulating. The gap between teleoperation and telepresence is then evident: it might be better (in terms of goals, usability and performance) to allow the operator to be fully immersed in the task,

[2]Instead of fully disengaging the master from the slave, it is possible to star with an arbitrary scaling. The user can move around and select to zoom-in (lower scaling) or zoom-out(higher scaling). Going to a higher scaling enables the user to reach previously unreachable points and then zoom into them to perform fine manipulation.

not only visually perceiving, but mechanically feeling what is happening in the remote environment by including haptic/force feedback to the operator performing a task.

However, as in any other physical and mental activity, an operator might become tired if all commands have to be issued in a very basic form all the time. A way to relieve the operator from part of the burden is to include some degree of automation and to design a proper user interface. These should not tax the operator's performance but help to perform the task faster and/or with better quality.

For this implementation Cartesian to Cartesian mapping was selected because the preliminary trials using the joint space to joint space mapping did not yield convenient results and a large amount of training was needed just to become aware of the mapping with no force-feedback rendering.

4.2.1 Clutching

Clutching is the most widely used approach when using asymmetric interfaces for commanding position. This scheme shifts a reference point and then uses a scaled command relative to this "offset" position to calculate a command for the slave robot in the teleoperation setting. The name makes reference to the way a mechanical transmission is decoupled from the motor supplying movement to it; in this case the master is decoupled, issuing the commands, from the slave, reproducing these commands. While the master is disengaged the movements performed by the operator are used to offset the reference point for the system. Procedure 3 gives the pseudo-code for the clutching master-slave motion coordination scheme.

The clutching master-slave motion coordination scheme is event-driven; *clutching* and *unclutching* are the events that happen when the operator commands the system to decouple or to recouple the master to the slave. While the master is decoupled from the slave, the system becomes *clutched* and no new position calculations are sent to the slave, thus it does not move. While the system is not clutched, the position commands are issued adding an offset calculated when the system was re-coupled. Moreover, while using this master-slave motion coordination scheme, the user only receives force feedback when the slave is moving.

4.2.2 Differential-End-Zone

Another master-slave motion coordination scheme for asymmetric settings is to let the operator issue both rate and position commands. The position on the master is used both to

Procedure 3 Clutching Algorithm

Input: $position_{previous}$ {the master's previous position}
Input: $position_{current}$ {the master's current position}
Input: $offset_{c}urrent$ {The anchor's current position}
Input: $scaling$ {The scaling factor master-to-slave}
Output: $output$ {The resulting position sent to the slave}

```
1: loop
2:   if clutching then
3:     position_previous ← position_current
4:   else
5:     if unclutching then
6:       offset_current ← offset_current + scaling × (position_current − position_previous)
7:     else
8:       if ¬clutched then
9:         output ← offset_current + scaling × position_current
10:      end if
11:    end if
12:  end if
13: end loop
```

command the slave's position from an offset reference, and to shift this offset reference. At any given time only one command is issued. A very similar approach is presented in [66]

When presented with a boundary and the point of interest *i.e.,* the point of manipulation, is beyond reachability, it is natural to push this boundary until the point of interest is within reach. It can be seen as follows: the robot can operate within a box of a given length, depth, and width but the point of interest is outside of that box. The operator needs to push and therefore offset that box until the point of interest is within the volume where he or she can manipulate. Pushing the box does not change its form nor its dimensions, it just shifts the space reachable by the local commands.

The differential End-Zone controller works in an analogous way. A fixed volume is defined such that it is contained within the "reachable" space of the master and leaves some margin before reaching the limits of the device. This fixed volume is then defined and centered in all three dimensions within the reachable workspace. The space is then used to render commands for pure position control. When the control point reaches the boundary of this defined volume, the position commands of the master are used to move the anchor's position. The commanded rate depends on how much penetration is there from the control point into the outer space or **end-zone**. This gives the effect of "pushing/shifting the box". While pushing on the boundary, the box will keep moving with the same direction as the normal to the boundary region. Procedure 4 gives the pseudo-code for this master-slave

Procedure 4 Differential End-Zone Algorithm

Input: $Limits \leftarrow Init\ Limits$ {Limits of the current workspace}
Input: $position_{current}$ {the master's current position}
Input: $offset_current$ {The anchor's current position}
Input: $scaling_{position}$ {The position scaling factor master-to-slave}
Input: $scaling_{rate}$ {The rate scaling factor master-to-slave}
Output: $output$ {The resulting position sent to the slave}

loop
  if $position_{current} > Limits_{superior}$ **then**
    $offset_{current} \leftarrow offset_{current} + scaling_{rate} \times (position_{current} - Limits_{superior})$
  else
    if $position_{current} < Limits_{inferior}$ **then**
      $offset_{current} \leftarrow offset_{current} - scaling_{rate} \times (Limits_{inferior} - position_{current})$
    end if
  end if
  $output \leftarrow offset_{current} + scaling_{position} \times position_{current}$
end loop

motion coordination scheme.

The operator needs to be aware if the boundaries have been breached by the control point in every time-frame and then shift the control point accordingly . The main advantage of this controller is that the events are not raised directly by the user but by a natural control action, simplifying the user interface. Special considerations have to be taken to guarantee that the full workspace of the slave is reachable.

For this controller to be effective, some feedback should be given to the user about the proximity to the zone where the commands change from position to rate. An unnoticed change from position based to rate based control might result in undesired behaviors in the slave side, *e.g.,* bumping into the environment.

The name of the controller comes from the idea of using the zones closer to the upper and lower limits of the workspace, or end-zones, to control the differential value of the position (rate).

4.2.3 Position/Rate Switch

The third master-slave motion coordination scheme uses a switch between issuing position or rate commands. The user then can trigger the change between issuing relative positions or a rate to change the reference point. This scheme can be thought of a hybrid between the two previous schemes (clutching and Differential End-Zone).

Position/Rate Switch works by issuing a relative position control signal in the same way as in clutching but also switching the controller to give rate commands to shift the anchor.

Procedure 5 Position Rate Switch Algorithm

Input: $position_{previous}$ {the master's previous position}
Input: $position_{current}$ {the master's current position}
Input: $offset_current$ {The anchor's current position}
Input: $scaling_{position}$ {The position scaling factor master-to-slave}
Input: $scaling_{rate}$ {The rate scaling factor master-to-slave}
Output: $output$ {The resulting position sent to the slave}

loop
  if *Rate Control* **then**
    if $Switch_{Rate}$ **then**
      {Current position as offset reference and attraction point for force feedback}
      $position_{previous} \leftarrow position_{current}$
    end if
    $offset_{current} \leftarrow offset_{current} + scaling_{rate} \times (position_{current} - position_{previous})$
  else
    {Position Control}
  end if
  $output \leftarrow offset_{current} + scaling_{position} \times position_{current}$
end loop

The user can raise an event in the same way as in clutching, in this case the event records the current position and calculates a rate used to shift the reference point. The pseudo-code for this scheme is outlined in procedure 5.

This controller is also event-driven. When the user commands to switch to rate control, the current position is recorded to use it as a second reference point to calculate the rates to shift the anchor. In the same way as the Differential End-Zone approach, the controller is always engaged and force feedback is always fed to the user. In the same way as in Differential End-Zone, the user should receive some additional feedback when using the rate mode as it is very easy to diverge from the point of reference used to calculate the rate.

4.2.4 Segway Rate Control

The previous subsections deal with the control of the manipulator, yet the mobility remains to be addressed. There are two main approaches to command a mobile manipulator. One considers the robot as an integrated kinematic chain, while the second approach takes the mobile base and the arm as two independent systems. The latter approach was selected as a first approach in this work while the former is left for future work. The operator can only control the mobile base or the arms but not both at the same time, *i.e.,* while commanding the arms, the mobile base remains static and *vice versa.*

The easiest and most simple controller for a mobile base is a car-like commanding ap-

proach providing a twist (combination of linear and angular velocities). In the case of a non-holonomic, *i.e.,* car-like, mobile base it is sufficient to provide a longitudinal (front-back) rate and a turning (left-right attitude) angular velocity orthogonal to the plane. These two are analogous to the throttle/gearbox and steering wheel in a car. For a holonomic mobile base this motion coordination scheme should be extended to provide the base with three inputs: rotation (clockwise or counterclockwise), stride (move left or right), and advance (move forward or backward).

A way to provide this car-like control is to encode or map the position on the master device to provide a twist. Usually the master device will have more degrees of freedom than those required to control the mobile base, thus simplifying the interface is not only possible but necessary.

In our case, the working of this controller is simple. The master's current 3-D position is used to form a vector from the origin. The resulting vector is then projected into a horizontal plane (x-y parallel plane). The component of the vector along the front-back aligned axis is taken and multiplied by a rate factor which will provide the front-back velocity for the mobile base. For the turning rate the component aligned with the sideways axis and the direction of going forward or backward are used. If going forward, the directions of left and right are taken without modification, but if the user is commanding the robot to go backward and adding an angular velocity, then the heading is inverted; if the user commands the robot to go back and to the left, the turning rate added is to the right. Because of the linear motion backwards, the mobile base effectively backs to the left (same case for the opposite direction). The pseudo-code for this controller is presented in procedure 6. Roughly, the sideways aligned component is analogous to the steering wheel in a car while the front-back aligned component is the gearbox & throtle.

A safety switch is always validated on every control iteration. This is called a Dead man's fuse and is used when the control of a device is given in rate. This is a safety feature that keeps the controller decoupled from the slave unless when needed. Leaving the controller engaged would increase the chances of issuing commands that could result in collisions.

4.3 Haptics and Force Feeback

In the current setting force feedback is scaled with the inverse of the factor used for the positions (Figure 4.2). This way when doing fine manipulation (Figure 4.2(b)) the forces

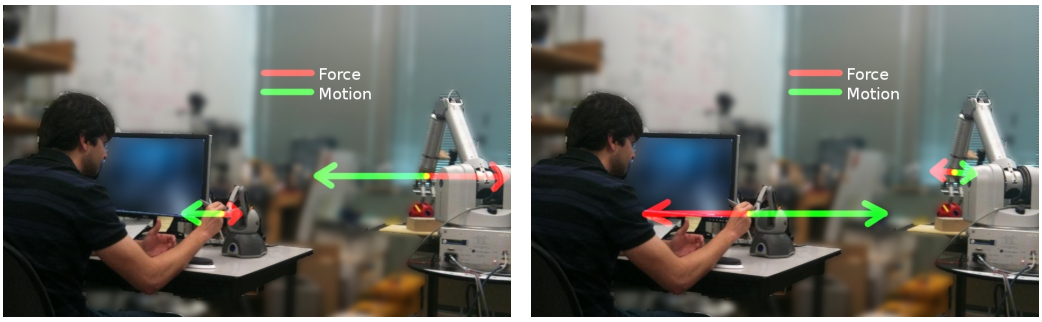
Procedure 6 Twist control for Mobile Base

Input: $position_{current\ on\ xy\ plane}$ {Only front-back and sideways components are taken. Height component is ignored.

Input: $scaling_{rate}$} {The rate scaling factor master-to-slave}

Output: $output_{twist}$ {Twist to move the mobile base}

1: **loop**
2: **if** $Safety_{ON}$ **then**
3: **if** $Front$ **then**
4: $Velocity_{angular} \leftarrow mapping_{positive}$
5: **else**
6: $Velocity_{angular} \leftarrow mapping_{negative}$
7: **end if**
8: $output_{twist} \leftarrow scaling_{rate} \times position_{current\ on\ xy\ plane}$
9: **end if**
10: **end loop**



(a) Rapid Traversing (b) Fine Manipulation

Figure 4.2: The implemented Force Feedback keeping impedance perceived by the operator unchanged. (a) Small master motions to large slave motions; large slave forces to small master forces, (b) Large master motions to small slave motions; small slave forces to large master forces

are exaggerated which in turn aids the operator to gain situation awareness and improve the overall accuracy of the motions. The latter is achieved because the inertia of the slave is also scaled by this factor thus the dynamics of the arm are felt in the force-feedback rendering. The heightened inertial feeling prompts the user to naturally avoid rapid motions. When doing large space traversing, *i.e.,* when small motions on the master are translated to large displacements in the slave (Figure 4.2(a)), the forces from the slave are rendered smaller on the master's side. These two settings do not change the impedance perceived by the operator mainly because the ratio of force and motion remains the same as both motion and forces are scaled up or down equally if seen from the operator's perspective.

The software haptic helpers, which will be introduced in the following section, and other haptic cues sit on top of the forces rendered. Forces due to interactions of the arm

and the remote environment have to be present at all times. Haptic helpers are used as aids to restrain degrees of freedom or confine the arm to a particular path or workspace; they should be added on top of any force feedback, as these are not providing information about the remote environment but only helping the user perform a task.

4.4 Software Helper Control Routines

A way to build some autonomy into a system is to automate gradually the process to do a task. This gradual automation should be done component-wise while keeping the automation elements as simple and basic as possible. Usually this means building primitives which take care of common behaviors, and/or reduce the error or variations introduced by the human operator while preserving the transparency of the decision making process.

The motions which can be used to build up automatic behaviors which can be classified under supervisory control rather than automation of the task. The components should be general enough to be used for any task. Automation of a task has been done several times in the past. The purpose of these automations are particular to the task rather than building a tool that can be used for several other activities or be sequenced to achieve a desired manipulation behavior. The objective is to build tools that can help an operator to perform different tasks instead of providing a solution for a particular task.

A Software Helper Control Routine (or helper for short) is defined as a tool that helps an operator perform part of a task by relieving the burden of giving all the explicit orders to achieve a certain desired output. The tools can either limit the input, remove repetition, fix some values and/or sequence actions or parameterizations. It is possible to work around several variables by controlling particular behaviors reflected as positions, velocities, accelerations, jerks, or forces.

Helpers can be classified into two categories: The ones that work mainly on the slave's side (the robot) and the ones that work on the user input or master's side. The former are called manipulation helpers, and the latter haptic helpers.

4.4.1 On the Robot's Side: Manipulation Helpers

By automating the process of moving from a position to another, these helpers can largely aid an operator by relieving the duty of teleoperating large displacements and coarse alignments. These two stages of manipulation are usually burdensome due to the large time and attention they demand. When no fine manipulations are performed, attention is used by the operator in trying to keep the displacements uniform, the velocities steady, accelerations

within certain limits, etc. When doing coarse alignments, the user might not be getting close enough to a particular position.

Open loop movements are acceptable when dealing with large displacements only involving the motion of the arm of a mobile manipulator. When large displacements involve the mobile base as well, a way to close the loop is necessary. Approaches like visual servoing using toolboxes like [50] can be used to overcome the problem.

In the open loop case, the simplest approach is to record a position on the manipulator (recording the joint values for a particular configuration) and then issue a command to replay it. This particular configuration should be parameterized or previously recorded to then replay it. This simple routine was implemented and used in an experiment. The results are presented in the next chapter. In the implementation used, the operator records the current place as an anchor point with a long key-press on a computer keyboard. The recorded position is loaded as the new anchor point when the user replays the recorded point with a short key-press.

In the case of closed loop repositioning, more elaborate schemes using navigation or incremental approaches of visual servoing can be used. Since this approaches are more complex in their nature, they should always be supervised; they do not require the operator to keep full attention nor issuing commands to achieve the goal.

4.4.2 On the Operator's Side: Haptic Helpers

Usually when asked to draw a straight line a normal person would think of using a ruler or any other "straight" aid to guide a pencil or a pen. The stylus leans against this aid to trace the line. It is natural to use physical restraints to limit the motion in certain degrees of freedom so that the task can be completed with more accuracy. The same idea is behind the haptic helpers except that the physical constraints are rendered by a haptic device instead of using an external body to limit the motions. In most cases, these helpers are used when doing fine manipulation tasks.

Geometric constraints such as path following or physicaly parameterized actions, *e.g.,* constant pressure on a point in a surface, are just two examples that can be obtained with haptic helpers. These target behaviors might be somewhat difficult to achieve with plain teleoperation. Haptic helpers relieve the operator by taking care of the controllable variables while allowing the user to still do the core of the task.

Line Constraint

Following line trajectories and making an operator operate in the proximity of a line or a point prevents the slave from interacting with the surrounding environment. Tasks like soldering, glue depositing, or even threading are greatly aided by including a simple primitive that enforces the operator to follow a line in space. These tasks have in common that the actions occur along a path. Usually this line can be parameterized. A straigh line only requires two points, for example. Further extensions are possible if more parameters are given and therefore more complex trajectories can be defined.

A line path enforcing routine was implemented and used in the drawing experiment presented in chapter 5. The user records a point A and a point B with key-presses on the keyboard. These recorded points are then used to fit a line passing through these two points. The line is used to render forces on the master device. When the user commands a position away from this line, the forces act like a spring pulling the control point to the closest point on the defined line.

Plane Constraint

Geometrically and physically constraining an operator not to pass a given plane in space has some uses when manipulation is supposed to be done over a surface which is not to be interacted with except for placing objects. This primitive can also be used to interact with a constant pressure or a constant distance to a plane. Tasks like assembling pieces, electronic board inspection, or any table-top activity can take advantage of such helper as the operator will not have to keep constant notice of avoiding collisions with the plane of interest as the system will be taking care of that avoidance with this plane restriction. This helper can easily be extended to use a parametric surface.

The plane constraint was also implemented and used for the drawing experiments presented in the following chapter. The user records three points, A, B, and C which are then used to define a plane in space. In the same way as the line routine works, a force proportional to the distance of penetration into the plane is rendered. There are three possible restrictions: Remaining within the plane, Staying above the plane, and Staying below the plane. All three options were implemented and made available to users. Each restriction could be turned on by pressing particular keys on the keyboard.

4.5 Discussion

Through this chapter the rationale behind a user-system interface used for teleoperation was introduced. When using a symmetric setting for teleoperation the master-slave motion coordination schemes are largely simplified as no scaling schemes have to be taken into consideration. Moreover, the large number of degrees of freedom and the likely redundant configuration increases the onus on the operator.

Since teleoperation is in itself a complex activity, Providing a suitable interface for the user to command the robot is paramount. It is also important to relieve the operator of unnecessary burdens by including some features that help automate certain parts of a task. Since scaling movements becomes necessary when dealing with an asymmetric setting, it is also necessary to be able to equip the command interface with a master-slave motion coordination scheme that allows the user to navigate the full workspace of the slave with ease.

Clutching is one of the most widely used approaches to command and shift the active workspace mapped from the master to the slave device. This approach engages and disengages the master from the slave and then uses the motions of the master to either command the local motions on the slave or to shift the anchor used to define the current "active" workspace.

Another approach is Differential End Zone. This scheme uses virtual limits from which the control signals are changed from position to rate. The change makes the reference point change in the same direction as the limit is breached. Special considerations have to be taken in order to limit the rate at which the reference point can be moved and also on how the motion near the limits of the slave's workspace are resolved.

Position/Rate Switch is a third approach combining the idea behind clutching and the permanent control offered by Differential End Zone. It allows the user to change between position and rate commands. Differential End Zone conserves the idea to move freely in an active workspace but it also includes the idea of moving the reference point using rate commands while keeping the control between master and slave always engaged.

Although the master-slave motion coordination schemes are an important part of the user interface, some other aids should be given to the operator. Software manipulation and haptic helper routines provide these aids by automating certain aspects of a task. Manipulation helpers can be as simple as an open loop motion to a previously visited zone in the workspace or as complicated as servoing to a particular configuration using closed loop

control (for example visual servoing). Haptic Helpers are usually implemented in the master side and serve the purpose of limiting or controlling certain freedoms of a task to to keep constant pressure, follow a determined trajectory, or restrict the manipulation to occur above a surface.

The product of the design presented in the previous chapter and the ideas presented in this one, were put to test with the implementation of some experiments involving user and case studies. The results for these experiments are presented in the following chapter.

Chapter 5

Manipulation Experiments

The following experiments have two purposes. 1) To provide proof of concept of the modularity of the system as a very desirable design characteristic allowing the system to be multipurpose and flexible. The configuration of the system and the modules to used can be changed quickly and on-demand. 2) To provide some insight and a preliminary study on the uses of teleoperation primitives to aid the user accomplish a task with more efficiency than with plain teleoperation. These experiments were designed to provide information to improve the vision setting, the control mappings, and the available helpers for the operator. These three areas are of particular interest as they provide insight into the fine tunning of the system setting and in general for later research activities.

The following sections overview the results of four experiments and a couple of cases of studies. First, the rationale on how the experimental setup was selected will be presented in Section 5.1. Two general tasks served as a sandbox/proxy for manipulation: Towers of Hanoi and Drawing which are described in Section 5.2. The experimental set up is outlined in Section A.3. Sections 5.3, 5.4, 5.5 and 5.6 explain the experiments and present the results obtained. Section 5.7 presents a couple of case studies where the system was used for mobile manipulation tasks. Section 5.8 concludes with a discussion on the generalities of the chapter.

5.1 Introduction

The focus for the experiments is set on haptic primitives for manipulation. As the experimental design was developed and some preliminary studies were performed, it was obvious that some system characterization was needed prior to conducting more elaborate experiments.

The configuration used in the robot is a one-arm mobile manipulator which for the sake

of simplicity will be used as a static manipulator for the experiments with human subjects and as a mobile manipulator for the case studies ran by the developer alone. The mobile manipulator will feature three cameras for most of the experiments.

A one-arm setting was used because it is sufficient to gain insight on teleoperation and its particularities. One-arm mobile manipulation is in itself a stripped-down version of a two-arm setting, but without the problems of arm interference, motion coordination, and redundancy resolution in the multi-robot sense. On-site (fixed) manipulation does not require to solve for coordinated motion of the mobile base and the arm, and, although this latter allows for more interesting manipulation options (throwing a ball[29], moving large loads using the body's inertia, large displacement pick-and place), it is an extension to an on-site setting. Tele-manipulation needs to be tackled down in parts; once insight is gained on how to do pure on-site manipulation, it follows that mobile manipulation is an extension to it.

Two cameras were placed behind the arm overlooking the scene and the end-effector, one on the left and the other on the right (See Figure 5.2 Overview Camera Pair). These cameras see the full scene in front of the robot from a left and a right perspective, resembling the human sight. No stereo vision setting was used at this time. This configuration keeps the end-effector and the arms in the field of view in the same way a person would manipulate on a table. The cameras were mounted to the mobile base so they provide a "static" overview of the front of the robot. In fact, they mimic the way humans are constituted with an "oversight" point of view over the manipulation workspace. The third camera is placed as an eye in hand and is attached to the arm's last link (See Figure 5.2 Eye In Hand Camera). This camera sees the end-effector and the immediate surrounding area of it. This camera is not static and is intended for visual servoing purposes, yet it can be used as if it were a *plane pilot's* point of view.

A fourth camera (See Figure 5.2 witness camera), independent from the robot, was used to record the trials from a bird's eye perspective. This camera is static with respect to the environment and the robot. It looks over the full testbed and the end-effector of the arm. This camera was later used to review the timing of the trials as well to identify placement errors through the trials. These videos can also be used to measure the exact planar offsets of the discs with respect to the stacks' centers. Figure 5.1 shows the Operator's station monitor with all the camera views displayed in the monitor.

Table 5.1 explains the different configurations of cameras used in the experiments. In the case of drawing, this camera was only used for the experiment's documentation.

Figure 5.1: The Teleoperation station showing the Phantom Omni (on the right), a keyboard and the monitor. The monitor displays (left to right and top to bottom) the left and right overview cameras, the eye in hand camera, and the bird's eye documenting camera. This minimal setting is used to teleoperate the full robot.

Camera Setting	Towers of Hanoi	Drawing	Case Studies
eye-in-hand focal length	3.5*mm*		
left focal length	6*mm*	12*mm*	6*mm*
right focal length	6*mm*		
top focal length	6*mm*		3.5*mm*
baseline left-right	100*cm*		50*cm*
setback from base center	65*cm*		70*cm*
height from base	100*cm*		
left/right centerline aim	convergent to center of testbed		parallel, 30° depression

Table 5.1: Camera configurations used in the different experiments and in the case studies.

When developing the system and trying to characterize its configuration. Some questions became evident:

- Where to place the video cameras and which cameras help operators have better performance?
- How effectively do manipulation helpers aid operators?
- Which motion coordination scheme is the more natural for operators?
- How effective are haptic helpers while performing a fine manipulation task?
- Is the mobile mainpulator suitable to perform tasks?

The questions served as a guideline to design the experiments and case studies contained in this chapter.

The first experiment (Section 5.3) tests different camera views available to the user while using the clutching control scheme. The second and fourth experiments, presented in Sections 5.4 and 5.6, test the uses of manipulation and haptic helpers while feeding the operator with all camera views and the clutching control scheme. The third experiment (Section 5.5) compares three different motion coordination schemes to command the arm movement while providing all camera views to the user.

Since the experimental setup is largely focused on users, most of the design is based on the ideas presented in [46]. Some training is needed to operate the system thus, all three experiments were run with the same eight participants to minimize the variability due to differences in experience. The author/developer participated as a subject in the tests to provide comparison between new and highly trained users. To minimize the effect of history and previous training over the experiment, a preliminary phase of general training was included. Depending on the user, this training could be from 15 minutes to one hour.

The overall experiments were conducted in three separate sessions. The first session was used for the general training and the experiment concerning the camera views. The second session, conducted in a separate day, consisted of testing the manipulation helpers and comparing different motion coordination schemes. The final session was set in a third day and covered the haptic helpers.

Two settings providing a proxy of common operations (*e.g.,* pick and place, fine manipulation, and tool operation) of manipulation were used. For pick and place, the towers of Hanoi served as a proxy combining long and short displacements. For fine manipulation

and tool operation, a drawing task was used as a proxy for other activities such as welding, material depositing, inspection and path following.

For the purpose of evaluation the accuracy and time taken to complete a task were recorded. The users were asked to fill out a NASA-TLX (Task Load Index) [39] survey. Time was used as the principal metric for the towers of Hanoi experiments and the changes in presure and length of errors were used to calculate a metric for the drawing accuracy. The NASA-TLX was used as a tool to understand how demanding was each task perceived by the users.

Important Remarks on the Experiments and Data Analysis

Appendix A shows the details about the experimental setup used through this chapter. Further details on how were the users trained, what they were asked to do and what were the variables which were recorded are contained in this appendix.

Through the chapter the results from the experiments are presented in error-bar charts. The graphs present the results per user. The data for the developer (highly trained operator) and the overall mean considering all test subjects are only included for the purpose of comparison. The decision on not using a standard statistical approach like a *t-test* or *ANOVA* to analyze the data was based on the limitation given by the amount on information gathered for the tests. For every condition, there is only three data points per user, having seven users yields 21 data points. Yet, different users showed different skill level when performing the teleoperation tasks. While some users used a very conservative approaches, others preferred to use the system more audaciously. This attitude was also reflected in the results for each individual. In turn, individuals were not having consistent timings between subjects. To overcome the effect of the attitude and exploit the few data points that were possible to recorded for each individual it was decided to make the comparisons user-wise and their data points were taken as a population (normal).

The data analysis was performed by first calculating the mean (μ) and standard deviation (σ) of a user's data "population" (size n) using:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - \mu)^2} \quad where \quad \mu = \frac{1}{n}\sum_{i=1}^{n} X_i$$

We defined an alpha $\alpha = 0.1$ which is equivalent to a 90% confidence interval. Using this alpha we can look up for extreme values that delimit the area under the normal curve corresponding to 90%: 1.645.

The two extremes of the error bars are calculated by solving:

$$limits = \mu \pm 1.645(\frac{\sigma}{\sqrt{n}})$$

This was done per user per testing condition.

It is important to note that because only 3 points were taken per user, in most cases their times were not that uniform. Each user's "population" is likely to have a large variance. This effect is expected to disappear if the number of test points per user and the number of test subjects were larger.

5.2 The Sandbox: Towers of Hanoi and Drawing

This section is dedicated to describe the two major settings used in the experiments. These settings could seem at first not related to real world applications, but by taking a closer look, they both use actions such as coarse and fine alignment, pick-and-place, path-following, etc., which are widely used in any kind of manipulation. Teleoperation tasks are largely specific to the application addressed; in a bomb defusing robot one would use lock opening and fine manipulation to cut wires or perform "delicate" pick and place. These same fine manipulation and delicate grasping could be used/learned/evaluated by a proxy task such as threading a needle.

The rationale to use a proxy instead of a real world application is that it provides a more generic setting than the application itself. This means that if one were to demonstrate pick and place, it is safe to say that the setting allows for flexibility and the robot can be used to pick-and-place any kind of object, or using any tool along a given path. Also, a sandbox case allows to discover general pitfalls which could be overlooked if taking a specific application instead. Even more, it makes it easier to discover conditions that must be taken into account regardless of the application at hand. Since the proxy can be constrained and defined at will, it is possible to remove/add restrictions without having to change the general setting.

Using a game, puzzle, or simplified task, such as the towers of Hanoi or doing a simple drawing, works in the same way as children learn through playing. It is possible to gradually increment the difficulty, constraints, and requirements of a given task while generating a more complete and generic solution which can be instantiated and improved for particular situations. A more detailed description of the two settings of the proxy tasks is presented in Appendix A. The towers of Hanoi focus on coarse and fine alignment, and pick and place; a picture of the test bed is depicted in Figure 5.2. Drawing focuses on tool-using and path planning; Figure 5.3 depicts of the setting used.

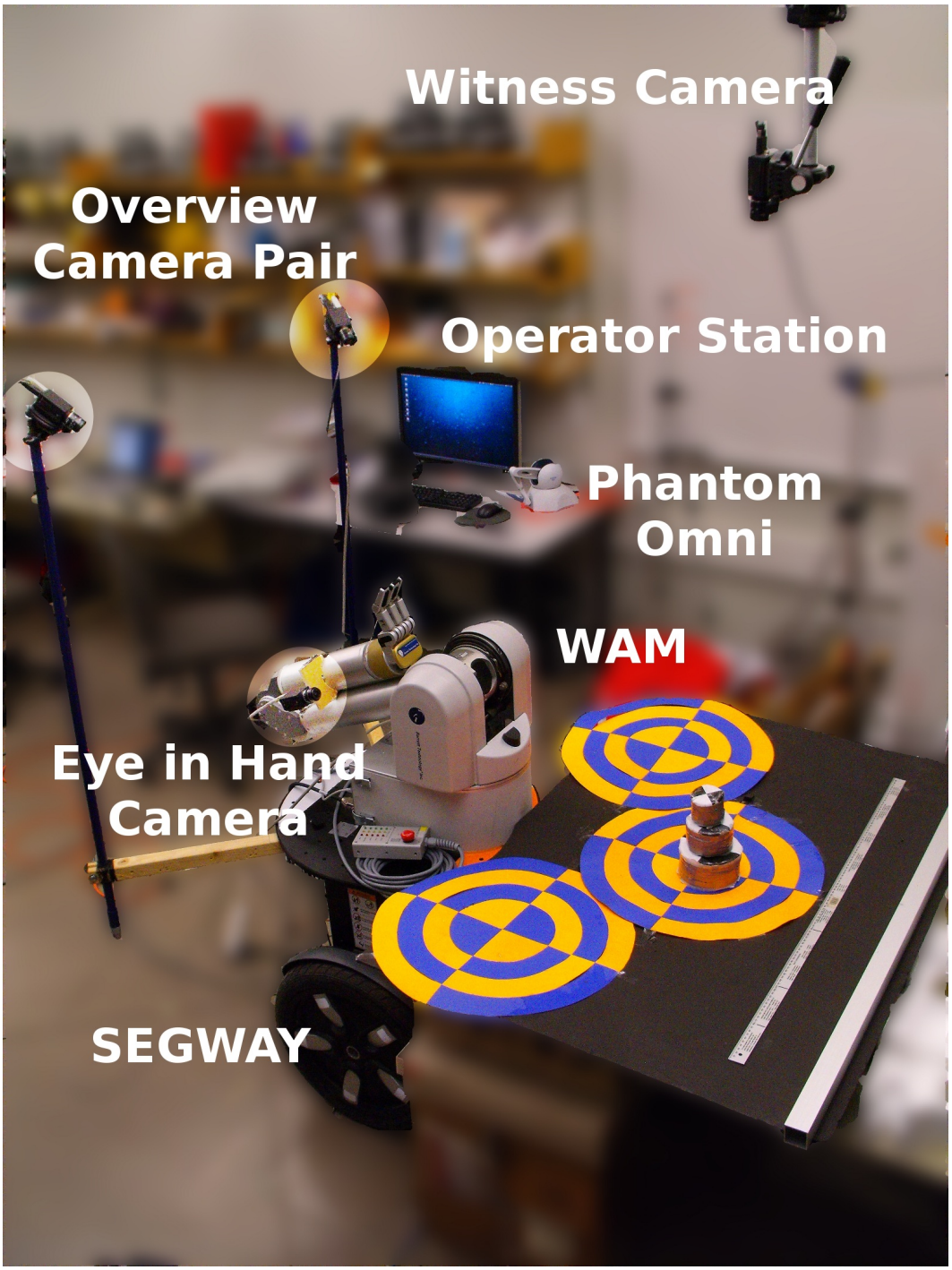
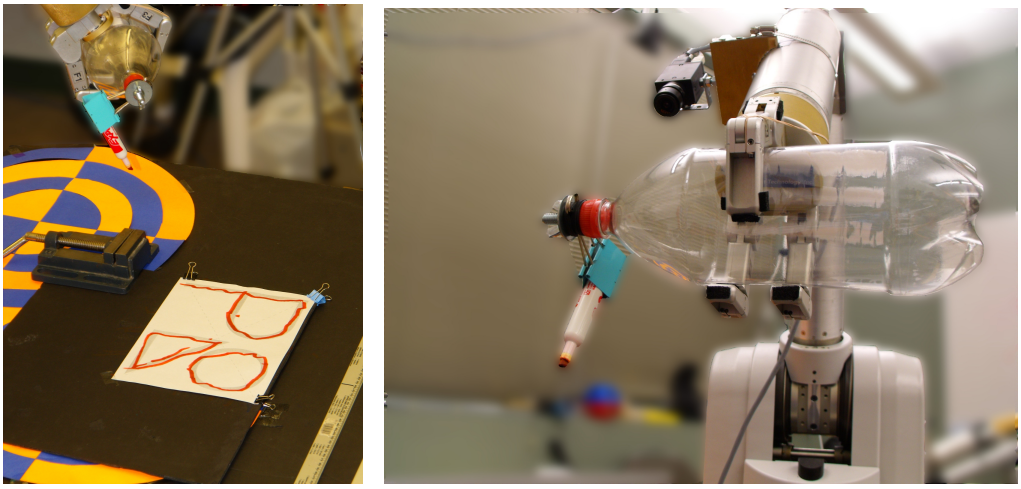


Figure 5.2: The picture shows the robot with all three cameras (two overlooking, left and right, and the eye-in-hand) and the testbed with the three places and the discs in the center stack. Additionally a top-viewing camera is shown. This camera was used to record the experiments in order to evaluate the precision of the placements. A 100*cm* ruler is shown to demonstrate proportions.



(a) Detail of the drawing area (b) Detail of the drawing tool

Figure 5.3: (a) shows the detailed view of the location of the drawing area and the end-effector holding the drawing tool. (b) shows a detailed view of the drawing tool used, which holds for a thick felt tip marker.

Setting	Value
Variable to test	Camera Feeds
Gold Standard	All three Camera views
Video Feed	Eye-In-Hand, Overview Camera Pair, and All three camera feeds.
Arm Control Scheme	Clutching (Fixed Scaling 1 : 2.5)
Helpers	None
Mobile Base	Disabled
Activity	Towers of Hanoi

Table 5.2: The configuration for "Selecting Video Camera Feeds" Experiment.

5.3 Selecting Video Camera Feeds

This experiment evaluated which video feeds from the cameras on the mobile manipulator provide better information for the operator. The hypothesis to test is: the cameras on an **overview camera pair** configuration allow the operator to finish the task faster than with the **eye in hand** for in-site pick and place task. Both approaches were compared to a third case, which will be taken as the gold standard: Providing the user with **all 3 camera** feeds: eye-in-hand and the overview camera pair. Table 5.2 summarizes the settings used in the experiment.

The time taken to perform a full instance of the towers of Hanoi was recorded. It was expected that the users would perform the task in more time when using the eye in hand or

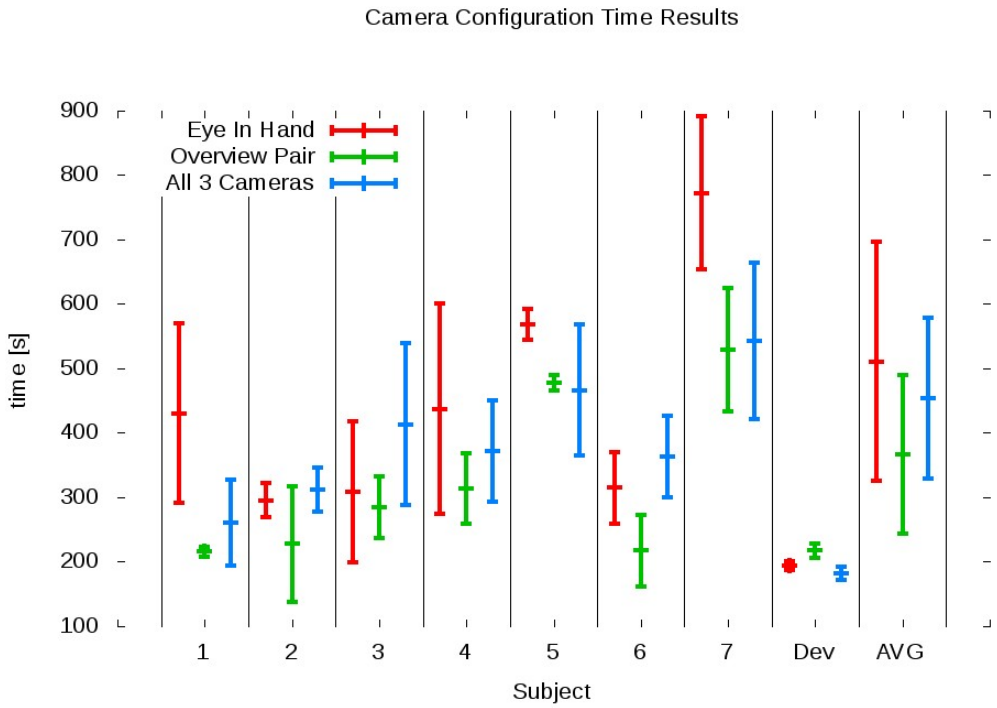


Figure 5.4: Times with 90% confidence intervals for the camera configurations experiment. Note that except for the subject (3), the rest of the users seem to have a preference either for the overview camera pair or the eye in hand configuration.

the overview camera pair configurations than when using all three video feeds because the available information to the user in these cases is less than when observing the scene using all three views. However, It was expected that the overview configuration would allow users to perform better than the eye in hand configuration.

5.3.1 Results

Observing the times the users took to perform the full activity (Figure 5.4), the overview camera pair was the configuration allowing the best times for almost all the subjects except for the case of the developer (Dev in the graphs). The preference of using the all-three-camera feeds was divided. In some cases the subjects were able to perform almost as fast as with the overview camera pair. Only one subject appeared not to have preference on the video feed.

Only for one subject, using all three cameras was the least preferred configuration. This could be because the amount of information to process is more when having more than one video feed. Presenting the user with more than two possible feeds to look at at any given time, and therefore more information to process, could not always be the best option.

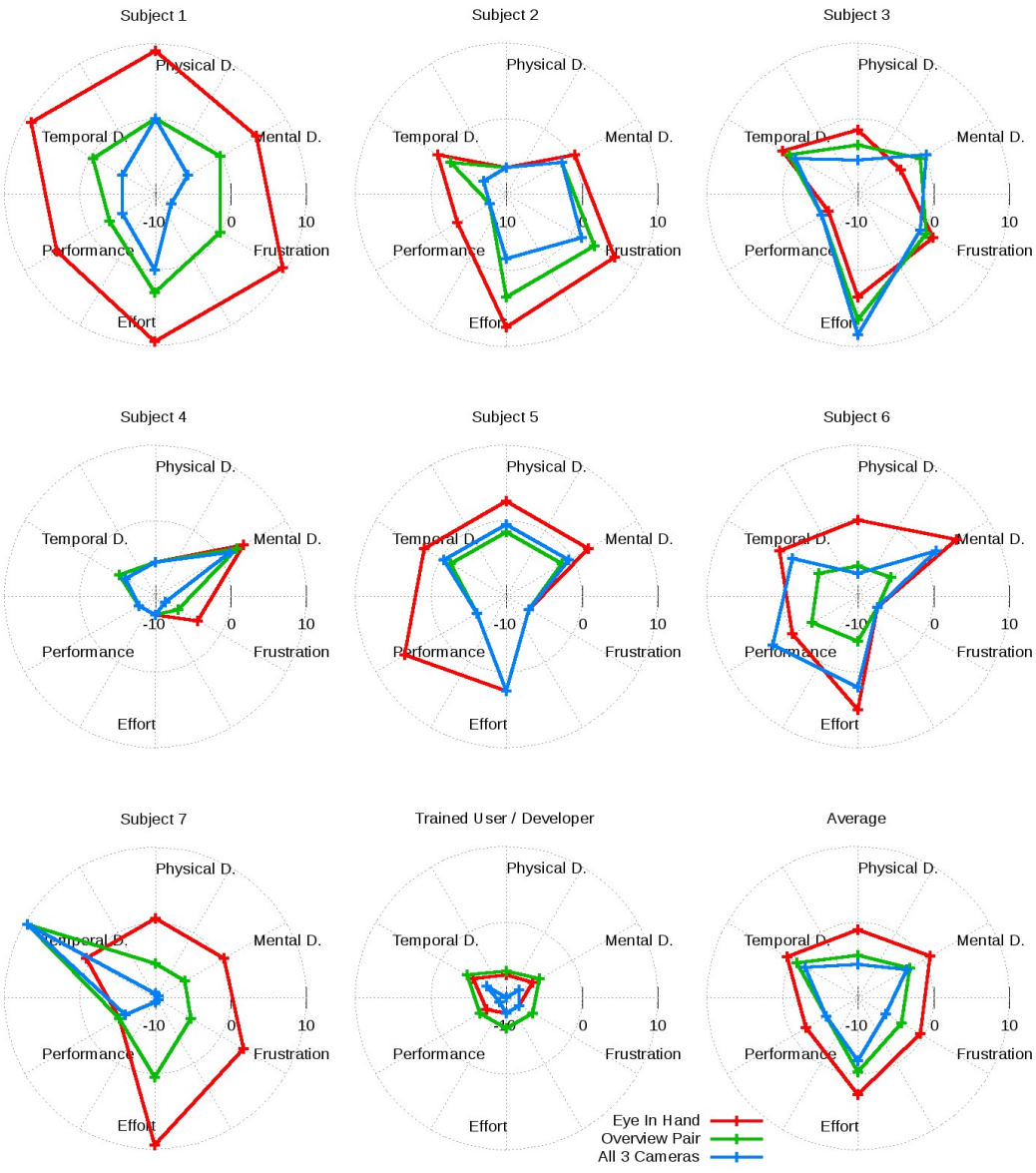


Figure 5.5: NASA-TLX raw scores for the different camera configurations tested in the Selecting Video Camera Feeds experiment. These raw scores were used to calculate the unweighed NASA-TLX score for each subject.

For all the other subjects, the times using the three camera configuration either resemble the times for the oversight pair or the eye in hand configurations. This could mean that the configuration at hand is used in the same way as the preferred configuration for each subject. Also, except for the subject (3), the rest of the users seem to have a preference either for the overview camera pair or the eye in hand configurations.

Looking at the NASA-TLX scores for the subjects (Figures 5.5 and 5.6) it is easily ob-

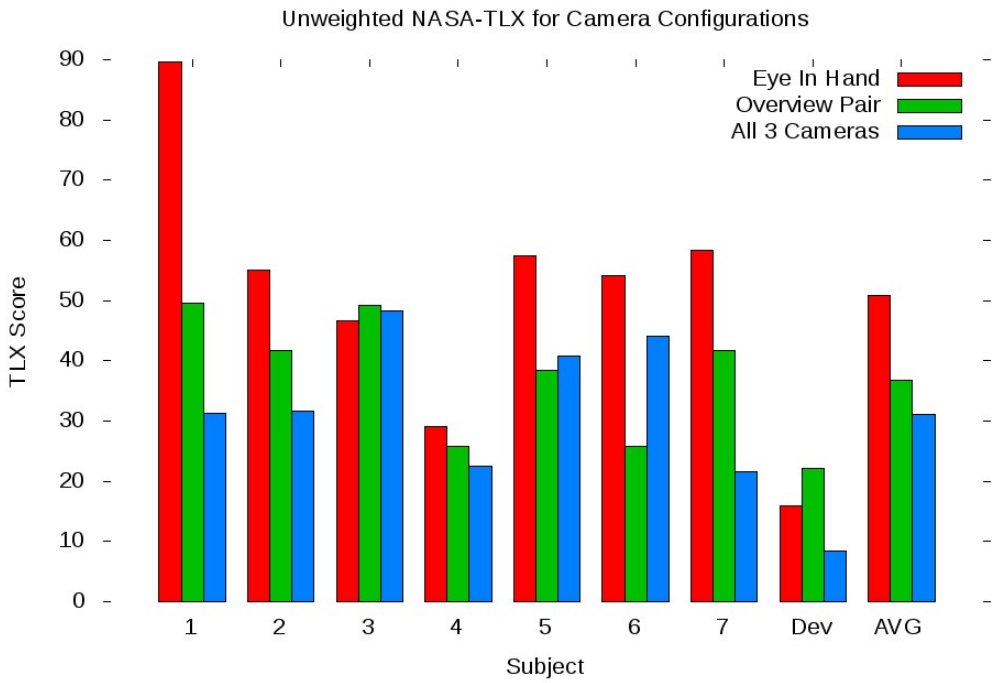


Figure 5.6: Unweighted NASA-TLX scores for the different camera configurations tested in the Selecting Video Camera Feeds experiment. For most of the participants the most demanding configuration is the eye in hand (except for the developer (DEV) and the person who did not have particular preference(3))

servable that for most of the participants the most demanding configuration is the eye in hand (except for the developer (DEV) and the person who did not have particular preference(3)). The main problem with this configuration is the lack of depth perception (lost due to the single point of view). The second most demanding configuration is the overview camera pair. Half of the participants said this configuration was less demanding than the eye in hand configuration because they could recover depth perception from combining information from the two video feeds. These same participants perceived the all three cameras configuration as the least demanding due to the amount of information they were able to get from that configuration.

Some users commented that the overview configuration, although natural, was too "stiff". In the future it would be interesting to reevaluate the use of the overview camera pair equipped with a pan-tilt unit to provide an "active vision" setting.

Participants 5 and 6 felt that all-three-cameras were more demanding than using the overview camera pair. This perception could be because the eye in hand view would distract them more than help them. The developer had a very particular classification. the overview camera pair was perceived as the most taxing, and this is because there are particular configurations for which there is incomplete information (when the arm is aligned with the principal axis of a camera). When using the eye in hand feed, some depth perception can be recovered by using the force feedback and the shadows that are projected in the environment.

As a general conclusion of this experiment, it is desirable to provide the operator with all the available information but it is necessary to train the operator to be able to use the information efficiently. This efficiency can be attained by training and by guiding the person on how to pay attention at different camera feeds depending on the task and the events happening. It would seem that the overview camera pair would give the better setting, yet because the lack of detail of the on-site fine-manipulation (given by the eye-in-hand feed), this setting might fall short of being the most useful. One of the main objectives in teleoperation is to be able to command the slave robot with the best possible decision. This is possible only if the most relevant information is given to the operator.

5.4 Using Manipulation Helpers

The objective of the experiment was to evaluate the usefulness of the place recording manipulation helper. This helper is used to record and replay set-points in the workspace of

Setting	Value
Variable to test	Manipulation Helpers
Gold Standard	Plain Manipulation
Video Feed	All three camera feeds.
Arm Control Scheme	Clutching (Fixed Scaling 1 : 2.5)
Helpers	Place Recording/Replaying, None
Mobile Base	Disabled
Activity	Towers of Hanoi

Table 5.3: The configuration for "Using Manipulation Helpers" Experiment.

the robot's arm. It can be used while performing repetition tasks or to keep configurations of interest/reset that could be useful while doing a manipulation activity.

The general settings for this experiment are presented in Table 5.3

From this experiment it was expected that the participants would be able to perform faster manipulation using the helper. The overall time needed to finish the activity is expected to be lower and the accuracy of the placements to be higher when using the helper. The time taken to complete an instance of the towers of Hanoi was recorded and then compared against the control conditions gathered before.

5.4.1 Results

In general, the manipulation helpers improved the overall times of completion for the users. As Figure 5.7 shows, in every case the times were shorter when using the helpers than when not using them. Only in one case, participant 4, the times show no difference (statistical significance). In every case, the times show less variance when using the helpers than when not using them. This is mainly because when not using the helpers, participants were prone to drop the disks while transporting them between stacking places or when depositing the disks provoking an incorrect stacking position.

When analyzing the data from the task load index (Figures 5.8 and 5.9), six out of eight participants felt that the task load was lower when using the helpers. Participant 3 pointed out that the load was higher because of mental demand, effort, and physical demand associated with remembering and replaying the recorded places. Participant 6 felt that the overall use of the helpers added load to the task as one had to remember settings and to set up the memory banks, record the places, and replay them to achieve the task. For these two participants is evident that the trade-off between controller complexity and task ease was not convenient.

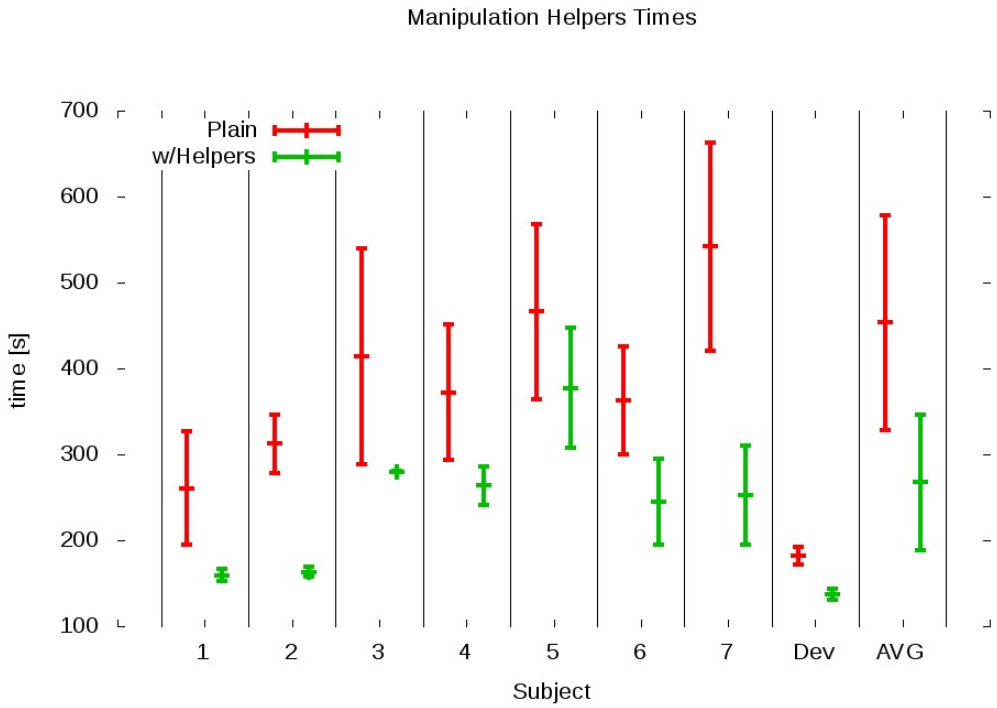


Figure 5.7: Times with 90% confidence intervals for the manipulation helpers experiments. The times were shorter when using the helpers than when not using them.

Most of the participants felt that the trade-off between complexity in the controller and the gained performance was advantageous. All participants regardless of their perception of the task's load, achieved better times by using the manipulation helpers. It is important to mention that none of the subjects knew there times before responding to the NASA-TLX questionnaire.

5.5 Evaluating Different Control Schemes

The experiment's purpose was to compare the different motion coordination schemes to command the manipulator. It was expected that Position/Rate Switch and Differential End Zone would allow the participants to complete an instance of the towers of Hanoi faster than with Clutching. Moreover, Differential-End-Zone would allow the users to finish faster than with Position/Rate Switch. The settings used for this experiment are summarized in Table 5.4.

The time for the trials was recorded and compared against each other and also to the already gathered control conditions. The expectation was that users would perform the fastest with the Differential End-Zone Scheme, slower with the Position/Rate Switch and

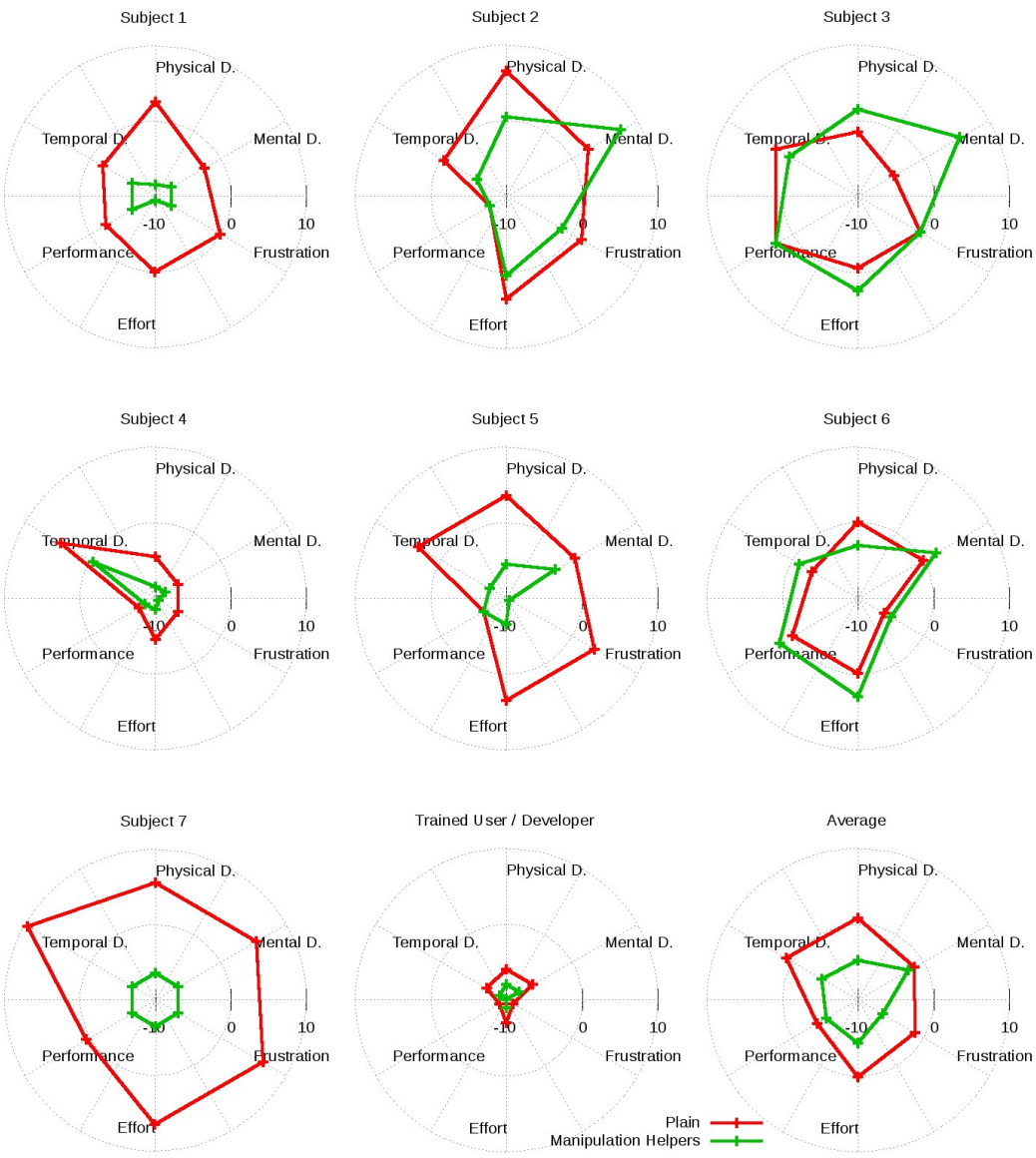


Figure 5.8: NASA-TLX raw scores comparing the use of Helpers and plain manipulation for the Manipulation Helpers experiment.

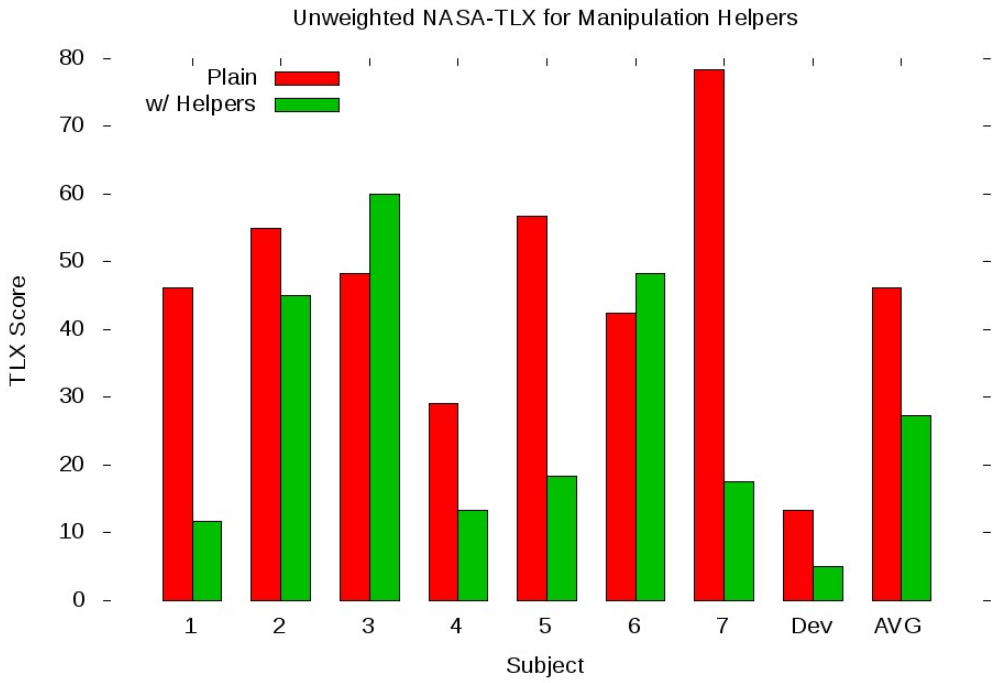


Figure 5.9: Unweighted NASA-TLX scores comparing the use of Helpers and plain manipulation for the Manipulation Helpers experiment. Six out of eight participants felt that the task load was lower when using the helpers. Participant 3 pointed out that the load was higher because of mental demand, effort, and physical demand associated with remembering and replaying the recorded places. Participant 6 felt that the overall use of the helpers added load to the task as one had to remember settings and to set up the memory banks, record the places, and replay them to achieve the task

Setting	**Value**
Variable to test	Control Scheme
Gold Standard	Clutching
Video Feed	All three camera feeds.
Arm Control Scheme	Clutching, P/R Switch, and Differential E-Z. (Fixed Scaling 1 : 2.5 for position)
Helpers	None
Mobile Base	Disabled
Activity	Towers of Hanoi

Table 5.4: The configuration for the experiment comparing the three motion coordination schemes.

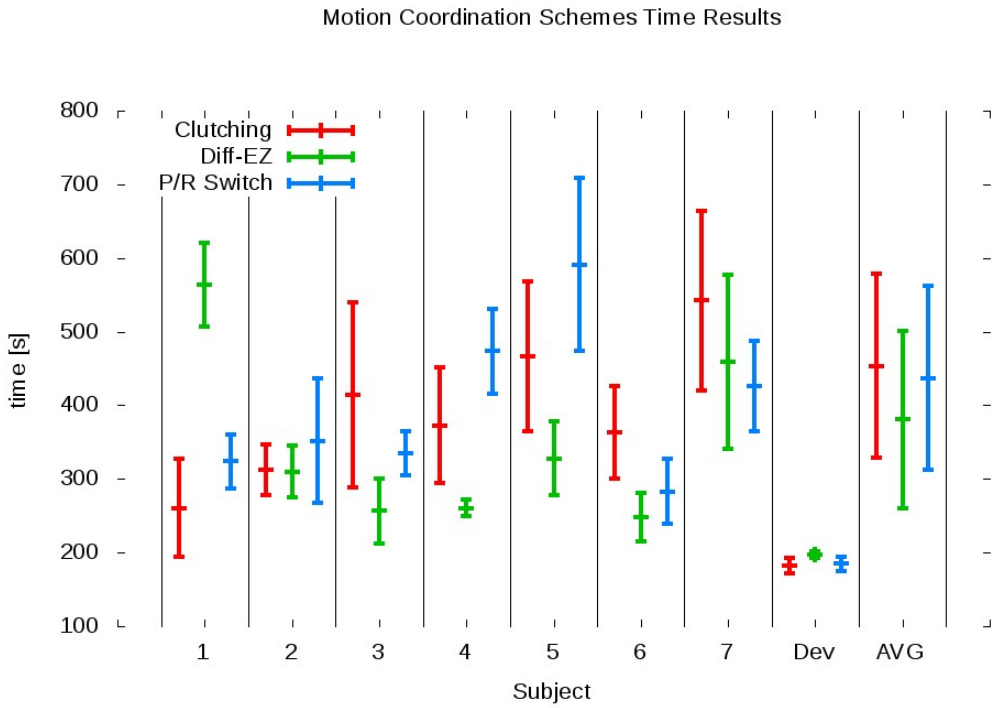


Figure 5.10: Times with 90% confidence intervals for the motion coordination schemes experiment. The best times overall were achived using Differential End Zone. Still, the results suggest that the preference of a scheme is user dependent.

finally the slowest with the Clutching Scheme.

5.5.1 Results

Figure 5.10 shows the times for participants and the different motion coordination schemes used. Except for subjects 1 and the Developer, all users achieved their best times using the Differential-End-Zone scheme; the exceptions achieved their best times with the Clutching scheme. For some subjects the schemes proved to be not significantly different (subjects 2, 7, and the Developer). In some way this lack of difference could be due to insufficient training (7), or to relative confidence and adaptation to the motion coordination schemes (2 and Developer).

Overall, users perceived the different controller schemes having a different load. Six of the operators perceived the Position/Rate Switch scheme to be the most onerous while the other two perceived clutching and Differential-End-Zone as the most taxing. It is interesting to see that while almost all users felt comfortable with the clutching scheme, some users preferred either the Differential-End-Zone while others the Position/Rate Switch schemes. Most users felt the physical demand, frustration posed by the Position/Rate Switch scheme

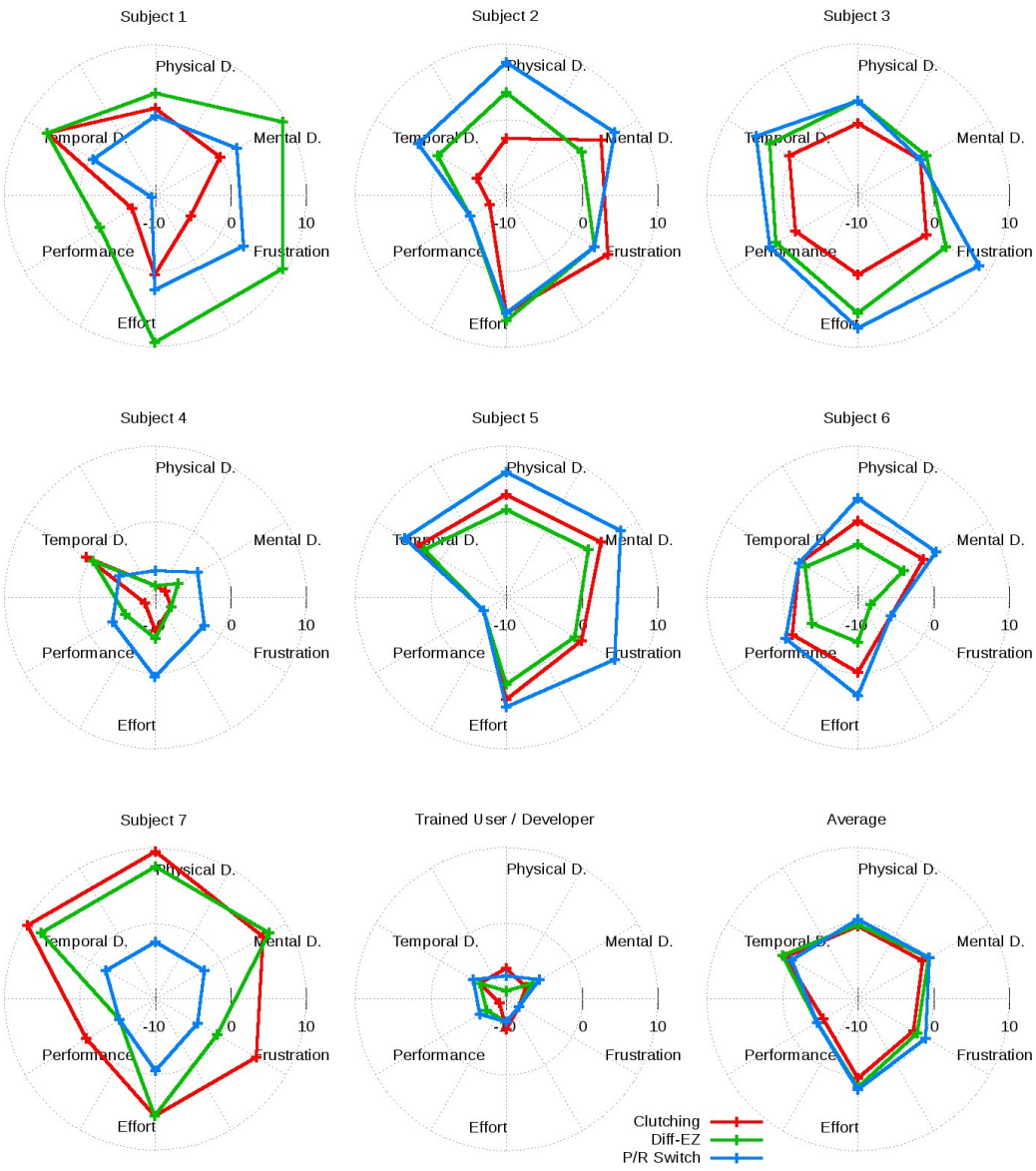


Figure 5.11: NASA-TLX raw scores comparing the different motion coordination schemes used.

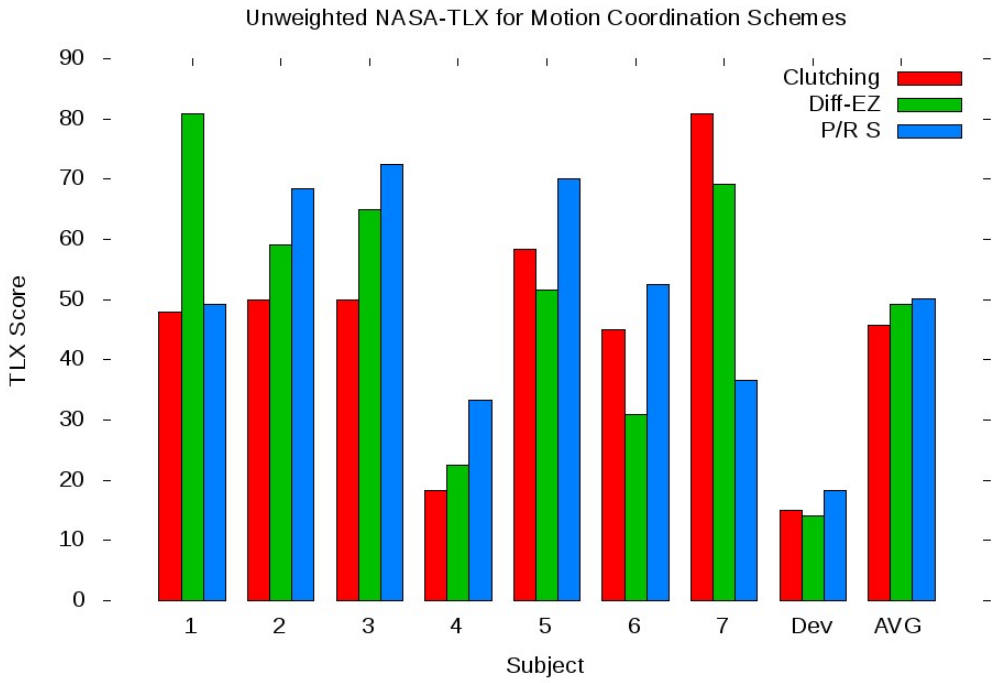


Figure 5.12: Unweighted NASA-TLX scores comparing the different motion coordination schemes used. as we can see from the unweighted NASA-TLX, clutching is the least taxing scheme, yet all of the schemes are almost the same from the point of view of task load.

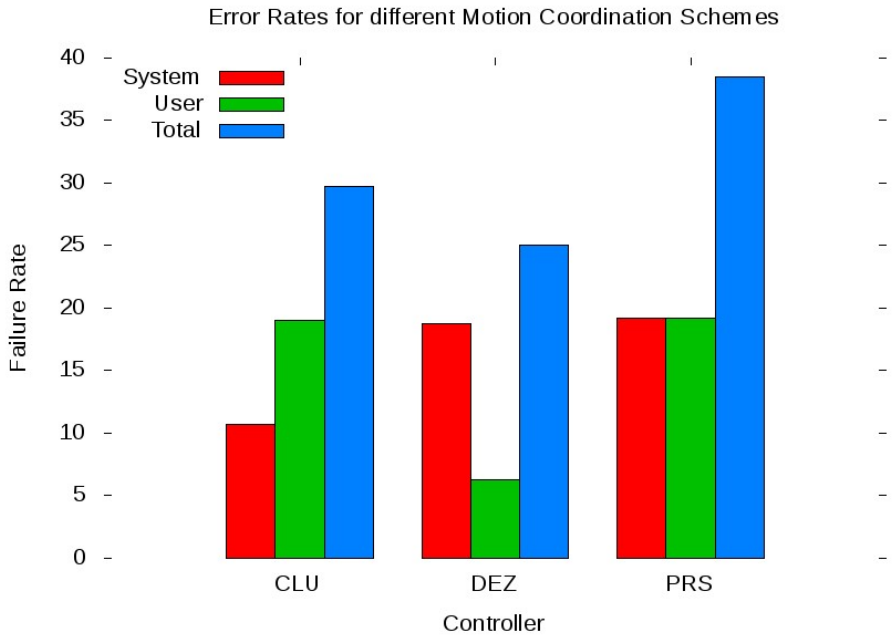


Figure 5.13: Error Rates for The different motion coordination schemes studies. Clutching and differential end zone have less errors overall. Moreover, differential end zone had the least errors provoke by user faults.

was higher than any other scheme.

Most of the users pointed out that the Differential-End-Zone scheme felt like the most natural approach, but it lacked the disengagement feature of the clutching scheme. The Position/Rate Switch was deemed the middle ground between the other two schemes, but all users felt that this scheme was non-intuitive and sometimes difficult to operate and predict in behavior. The clutching scheme is still regarded as the most stable scheme but it was tiring to shift the anchor when doing large motions. Several clutching events have to be performed to offset the slave from a start to a goal positions.

Clutching is still the most generic scheme, while the Differential-End-Zone seems the most natural approach. The latter is a promising avenue to develop more “natural” or ergonomic controller schemes for operators. It would even be necessary to test different tasks to evaluate the different controllers against different tasks, and prove if different schemes provide advantages for different kinds of tasks or activities.

The number of times the controllers became unstable, inoperable, difficult to use due to some force-feedback being rendered incorrectly, or when user errors occurred was recorded for comparison of the motion coordination schemes. Figure 5.13 shows the general error

rates for the three controllers. The bars depict the failure rates for system-based errors (instability or interoperability), user based errors (non-recoverable errors during the trials such as dropping pieces from the table), and the total failure rate.

It is evident that Position/Rate Switch has the most errors in any case. Users tended to make more mistakes because the change between position and rate, although voluntary, was sometimes not controllable or even provoked them to drop the stacking pieces outside of the table. Overall Differential-End-Zone had the lowest error rate, yet the system was highly unstable and became inoperable sometimes. Clutching seemed to be the most stable controller from the system point of view, yet most of the errors provoked by users were also made while using this scheme. Some operators observed that this was because of the interface itself and it was easy to mistakenly command the hand to open the gripper while the intention was to clutch the arm.

5.6 Using Haptic Helpers

This experiment provided information on how useful the haptic helpers are for the user. The main objective of this experiment, unlike the ones presented before, is to evaluate whether the helpers improve the accuracy of the operator while performing a fine manipulation task. The haptic helpers used in this experiment were the line and plane primitives. These helpers would be used to draw lines and they were also expected to aid in the drawing of circular trajectories on a plane.

It is expected that the traces drawn by the participants using the haptic helpers would be straighter, more even, and would have less variations from the original path than when drawing freehand. It was expected that when using the helpers, longer completion times will be recorded due to the overhead of parameterizing the helper.

The configuration for the experiment is presented in Table 5.5. Note that in this case the users are using a finer scaling in order to give them more control over the drawing process and allow them finer manipulation.

As presented above in Subsection A.2, the quantification is done with a metric of the accuracy of the drawing *i.e.,* how evenly pressured and close the traces commanded by the participant are to the actual paths requested. The time to complete the task was also recorded to document the trade-off between time and accuracy.

Setting	Value
Variable to test	Haptic Helpers
Gold Standard	No Helper
Video Feed	All three camera feeds.
Arm Control Scheme	Clutching (Fixed Scaling 1 : 1.1)
Helpers	Haptic Constraints: line and plane primitives; None
Mobile Base	Disabled
Activity	Drawing Patterns (P, Circle, and Triangle)

Table 5.5: The configuration used in evaluating the haptic helpers.

5.6.1 Results

Users were largely helped by the haptic helpers while drawing. All users performed the tasks with less changes in pressure while tracing the lines. Figure 5.14 shows the number of pressure changes recorded when using the haptic helpers and when using plain manipulation. In just one case, the difference was not statistically significant. It is worth noting that the variance of the recorded pressure changes for this case is high while the variance when using the helpers is considerably smaller. Less pressure changes while drawing means more even interaction between the robot and the surface of interest. In a real task, this could mean that there are less chances of making undesired collisions or high-force interactions with the environment.

As Figure 5.15 shows, the errors were almost the same when using the helpers than without. Only for subject 1 and the developer, a statistical significant difference was registered. This could be largely because the users were not trained sufficiently to use the helpers. An interesting observation is that although the errors were approximately the same for either case, the overall traces using the helpers were closer to the desired lines but the tracing was done with a regular offset from the intended line. When using the plane restriction, curves were also done with more continuity, but again with a constant offset from the target trace.

Regarding the times, in Figure 5.16, the task took longer while using the helpers and this was mainly due to the overhead of defining the restriction planes or the trajectories to follow. Overall, the increase in time is between 100% and 250%.

The perception of the users on using the helpers was divided. Half of the participants thought the helpers were helping them while the other half thought they were actually impeding them from finishing the task efficiently. In fact, two of the operators (subjects 2 and 4) were not certain whether the helpers were that different if using plain manipulation.

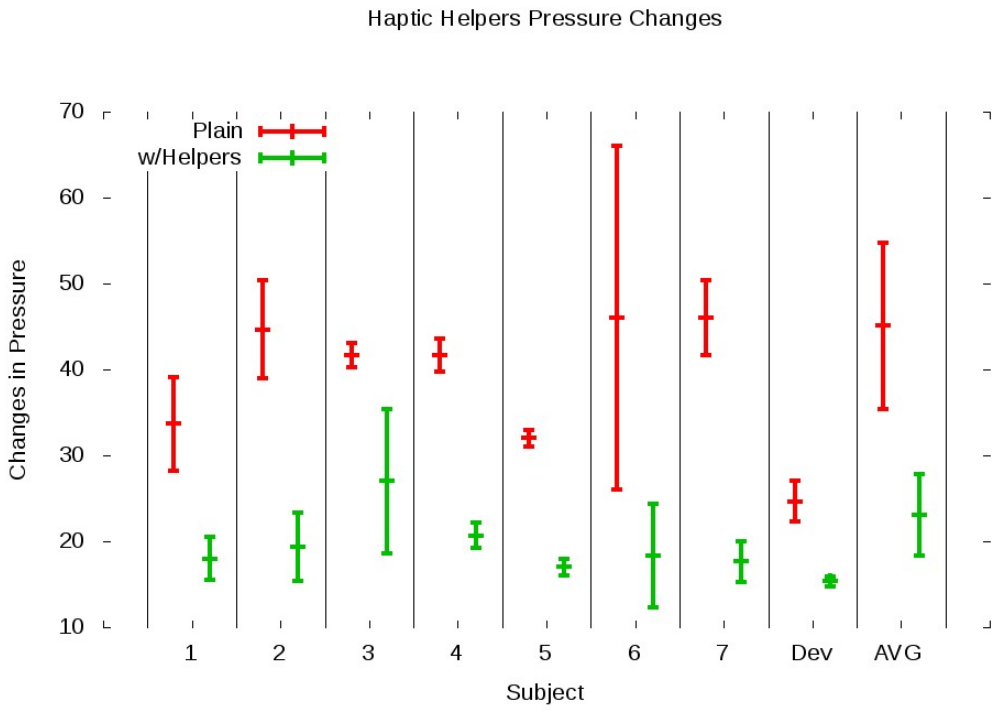


Figure 5.14: Pressure changes with 90% confidence intervals for the haptic helpers experiment. Users were able to draw with more constant pressure over the paper when using the haptic helpers (both line and plan restrictions).

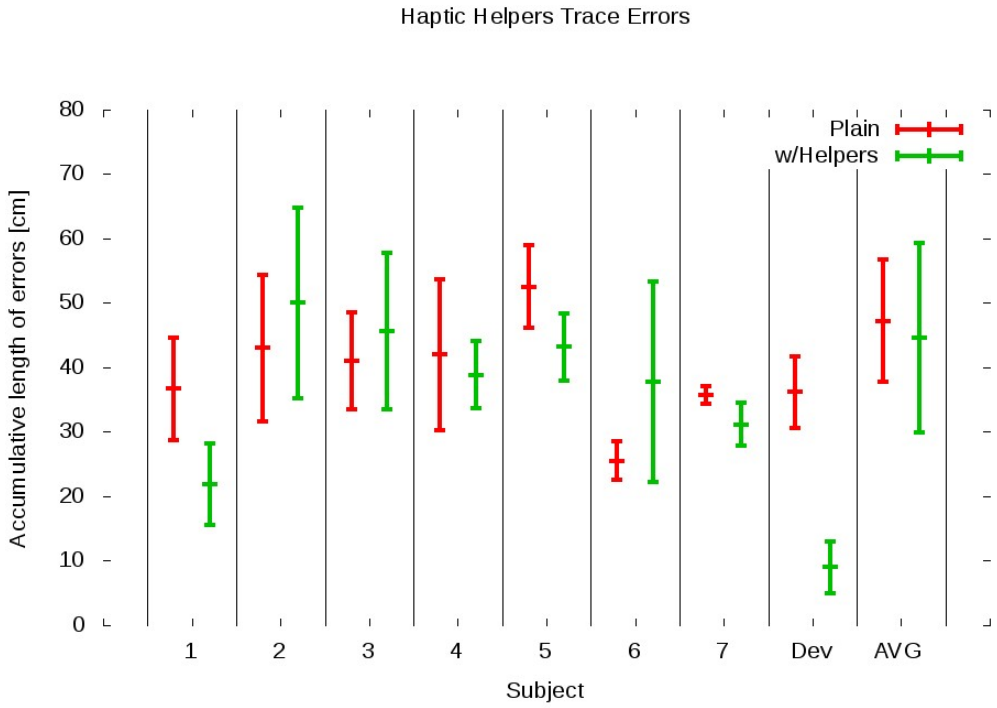


Figure 5.15: Length of error in tracing with 90% confidence intervals for the haptic helpers experiment. The haptic helpers did not perform as well as expected. In this case the amount of errors (counted as the times and length of a trace going out of the grey line), remained almost the same when using haptic helpers or plain manipulation. Only for subject 1 and the developer, a statistical significant difference was registered.

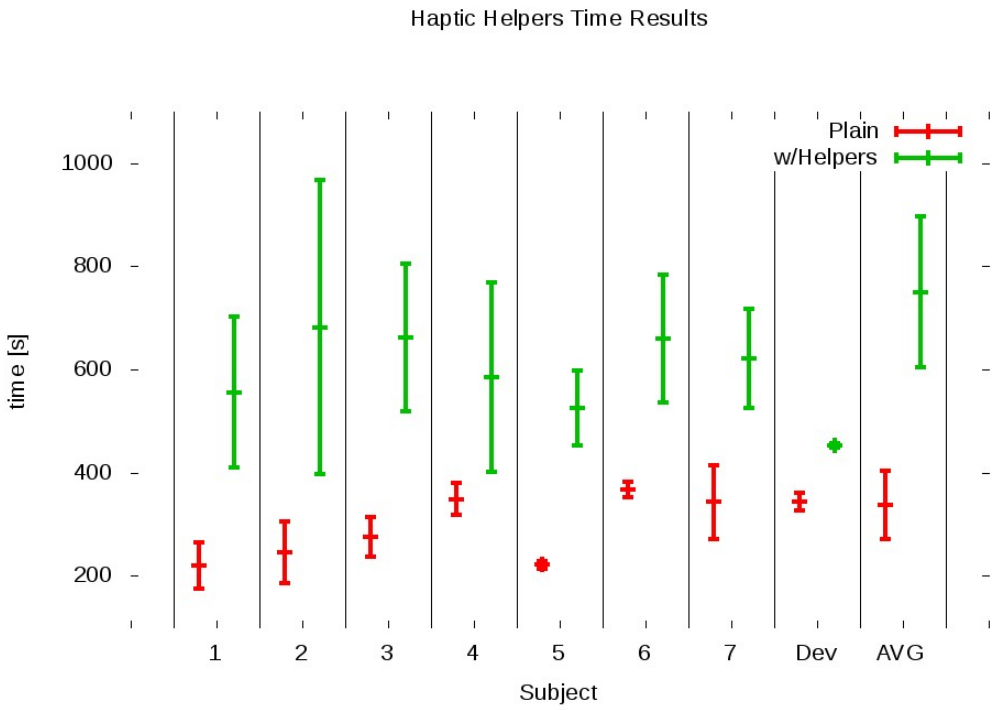


Figure 5.16: Times with 90% confidence intervals for the haptic helpers experiment. Times when using the haptic helpers were larger because they include the overhead of parameterizing the helper.

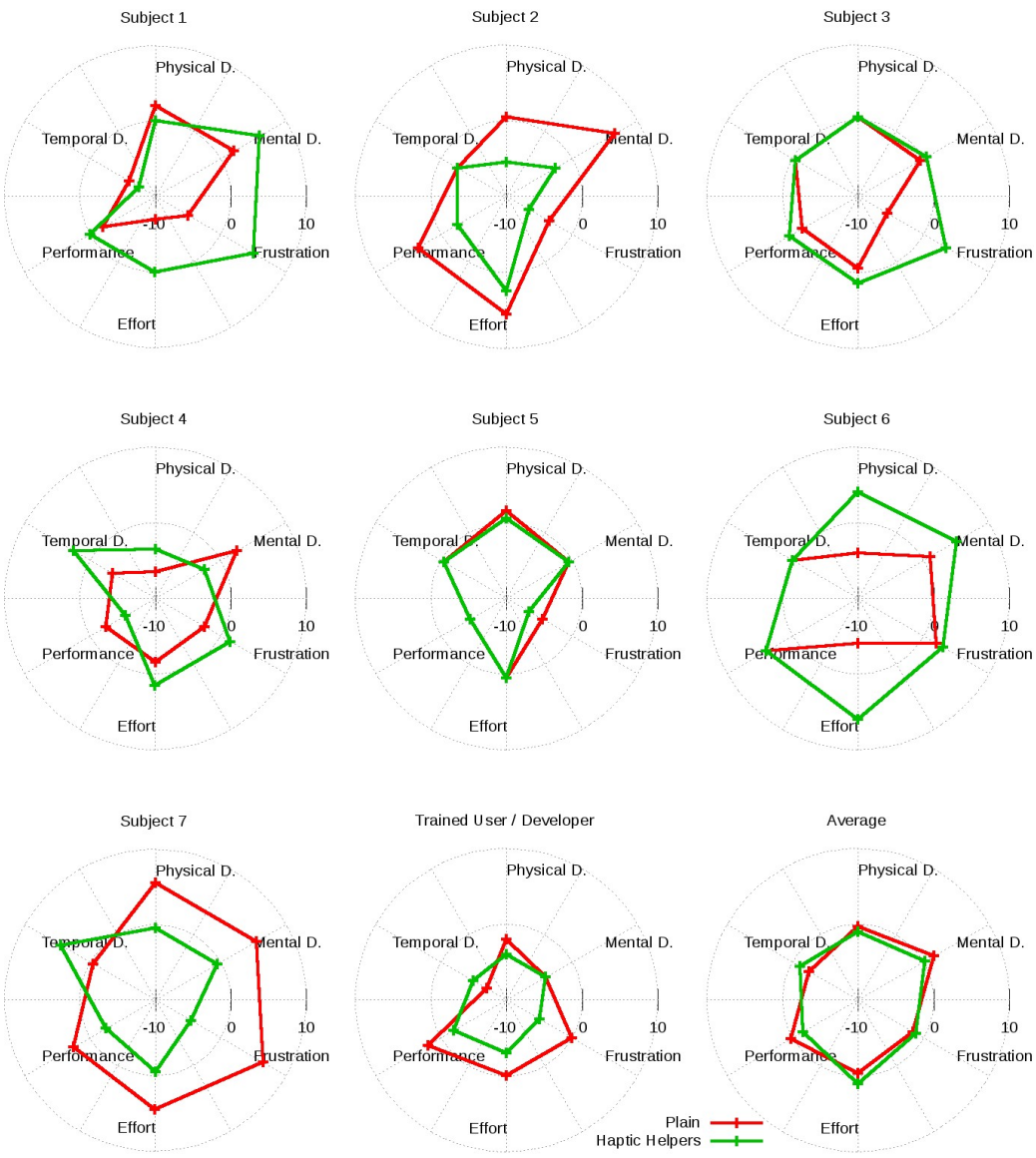


Figure 5.17: NASA-TLX raw scores comparing freehand drawing and Haptic Helped drawing

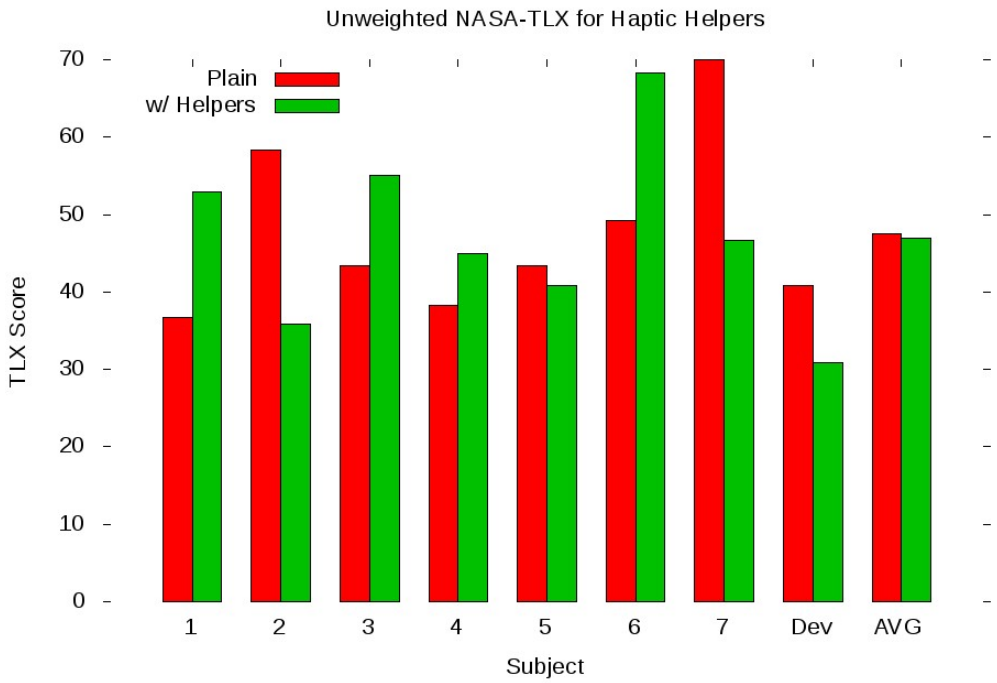


Figure 5.18: Unweighted NASA-TLX scores comparing freehand drawing and Haptic Helped drawing. Load perception when using the helpers or plain manipulation was divided. Some users felt that the helpers indeed reduce the onus while others felt that using the helper was more taxing.

They differentiate their preference based on the frustration the task gave them.

Overall, the haptic helpers facilitated the users to make even-pressured traces and therefore softer contact interactions with the environment. Although the overall number of errors was not improved, there is still room to test for these helpers while defining better primitives that could help attain better trajectory following. Giving the users more time to train and gain more confidence on using these helpers would also be desirable.

5.7 Mobile Manipulation Case Studies

The objective of these case studies is to demonstrate the capabilities of the mobile manipulator to perform a large-displacement pick and place task: large-displacement towers of Hanoi.

The first case study is an instance of the towers of Hanoi puzzle with the stacking places further apart from one another. This setting requires to move the mobile base to be able to reach the stacking places. The particularities and results of this case study will be presented in Subsection 5.7.1.

The second case study is a more day to day activity that has been proven to be difficult for robots: Opening doors. Teleoperated bomb-defusing robots usually have problems opening car doors and some mobile manipulators have problems navigating environments with closed doors. This task has received particular attention from the robotics community due to the complexity of the task involving grasping, planning, manipulation, etc. The objective of this second case study is to also demonstrate the capabilities of the system. The details are presented in Subsection 5.7.2.

The case studies trials were run only by the system designer/developer as commanding the robot for this kind of activity needs several hours of training and detailed knowledge of the system and its capabilities.

5.7.1 Revisiting the Towers of Hanoi

As said before, this case study is the mobile manipulation version of the setting used for the experiments described before in this chapter. The general setting was designed to test the capabilities of the system in a mobile manipulation context. The stacking places were placed far away from each other requiring to navigate the environment to finish the puzzle. Figure 5.19 shows a picture of the setting for the Towers of Hanoi with large displacements.

Three trials were run and the times were recorded using all the basic rules of the original setting of the puzzle. The instances used for the trials were selected in such a way

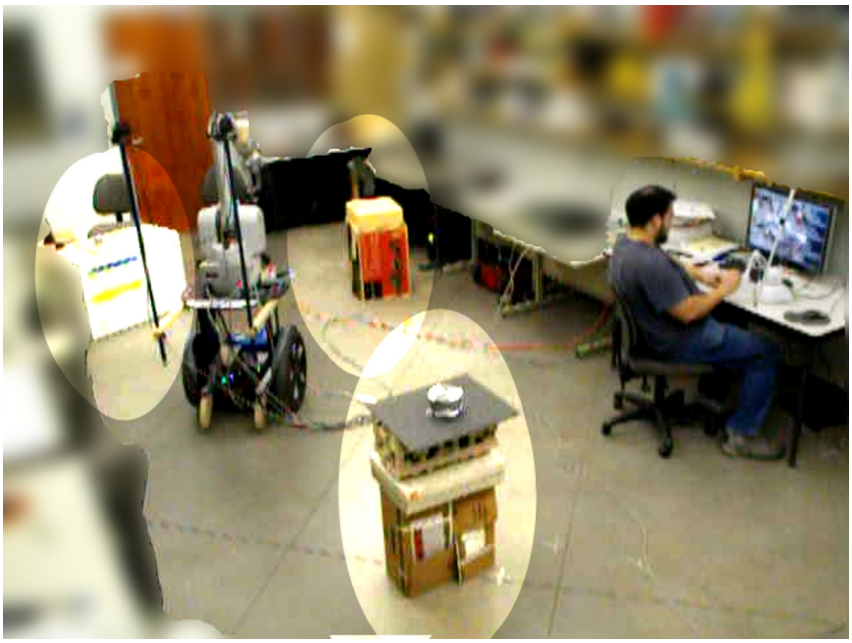


Figure 5.19: The setting for the Towers of Hanoi with large displacements. The three stacking stations are highlighted.

that the stack, at the end of one trial, would be placed in the starting position for the next. The times for the trials were: 24 *minutes*, 58 *seconds*, 25 *minutes*, 32 *seconds*, and 25 *minutes*, 53 *seconds*, yielding an average time of 25 *minutes*, 28 *seconds*. Surprisingly no errors were made such as bumping into the props or the limits of the room, or dropping pieces while manipulating or stacking.

Throughout the runs of the experiment, the clutching mapping was used for most of the time and other schemes were tried only in the first trial. The scaling used was increased and decreased several times in order to make more efficient movements while manipulating the pieces; selection of a finer scaling while aligning or unstacking the pieces was crucial for achieving good grasping. Coarser movements for rough-alignment were essential to traverse free-space faster. Also, the manipulation helpers were used to make rough height alignments.

5.7.2 Opening a Door and Exiting a Room

Opening a door has been an open problem in robotics for some time. It is known that opening doors is a problem that can be found in almost any task involving human environments. Opening doors can be used for environment navigation and also in security related applications such as investigation of suspicious packages in cars or bomb defusing. Figure 5.20

Figure 5.20: The teleoperated mobile manipulator in the initial setting to open a door.

shows the initial stage for this case study.

Three trials were run to open the door. All the trials started from the same initial configuration. The times to open the door, partially exit, come back, close the door, and come back to the original place were recorded. In general all trials were successfully completed without any system failures or restarting needed. The times recorded for the three runs were: 13 *minutes*, 25 *seconds*, 12 *minutes*, 32 *seconds*, and 14 *minutes*, 41 *seconds* giving an average of: 13 *minutes*, 33 *seconds*.

Clutching was used all the time because it provided position control in every instant. It was possible to attain better control over the proximity to the contact and manipulation points. Contacts and overall manipulation were done with extreme care to prevent any failure of the system and also to minimize the potential harm the robot. The haptics in this case were very useful to know when the robot was in contact with the door's handle and when it was turning it to open the lock. The haptics module also became useful when pulling the door open using the arm.

Camera Configuration	Average Time $[s]$	Average Unweighted NASA-TLX
Eye-in-hand	415.125	50.781
Overview Camera Pair	**310.542**	36.771
All-3 Cameras	364.042	**31.094**

Table 5.6: Averages over all the participants' times and NASA-TLX Scores for each camera configuration tested. Although the Overview Camera Pair yields overall faster completion times, the All-3 Cameras setting is the least taxing for the user.

5.8 Discussion

This chapter presented the experiments and the results obtained from a set of user studies. The participant operators commanded the manipulator (and a mobile manipulator in the case studies) to perform an activity. These experiments were designed around settings which sampled real-world generic manipulation tasks yet allowed for evolving condition and requirements. Future studies can be deployed and tested using the same proxy tasks.

Different experiments were conducted to evaluate the overall setting of the mobile manipulator.

The first set of experiments evaluated how different camera feeds affected the operators' performance. Despite the results not being as conclusive as expected, in a more detailed analysis, it is possible to conclude that having more information about the remote scene yields improved performance. It is necessary to train the operators, let them practice, and guide them on how to use the information that is being displayed (help them learn to discard non-essential information and notice the details). Table 5.6 gives a condensed summary of the results on the first set of experiments.

The second set of experiments showed how manipulation helpers could be used and evaluated how useful these were to speed-up the execution of a task. The manipulation helpers indeed helped the participants perform the task faster, but at the cost of demanding some more attention from the operator. Table 5.7 gives a summary presenting the averages for times and task load scores for all users.

Different motion coordination schemes for asymmetric systems were tested in a third set of experiments. Overall, the proposed schemes were not different from each other once the participants tried them. Regardless of this, and although clutching being the most generic approach, there is still room to design more ergonomic controllers which include different mappings. It is recommended for future studies to evaluate different tasks against different motion coordination schemes. Each approach might have some desirable behaviors which

Mode of Manipulation	Average Time $[s]$	Average Unweighted NASA-TLX
Plain Manipulation	364.042	46.198
Manipulation Helpers	**234.708**	**27.396**

Table 5.7: Averages over all the participants' times and NASA-TLX Scores for each mode of manipulation tested. Using the manipulation helpers both reduces the completion times and the perceived general load of the task. Despite of this, result, it is important to note that some users did feel that there was more attention needed when using the helpers than when not using them.

Control Scheme	Average Time $[s]$	Average Unweighted NASA-TLX	Failure Rate $[\%]$
Clutching	364.042	**45.677**	29.751
Differential-End-Zone	**327.917**	49.167	**25.00**
Position/Rate Switch	371.042	50.104	38.462

Table 5.8: Averages over all the participants' times and NASA-TLX Scores for each control scheme tested. Differential-End-Zone gave the overall faster performance and the less failure rate while clutching is still the less taxing for the operator.

could result advantageous depending on the nature of a task. The general results are summarized in Table 5.8

For the fourth and final user study, haptic helpers were demonstrated and tested. These helpers proved to be useful to minimize the high-force interactions between the slave and the remote environment. The haptic helpers aided the users to trace more even lines with constant pressure over the plane of interest. Although the helpers did not seem to help on the accuracy of the traces, they did improve the overall straightness of the lines drawn by the operators. Table 5.9 sumarizes these results.

Two mobile manipulation case studies were presented. In the first case, the task was an

Drawing Mode	Average Time $[s]$	Average pressure changes	Average Error Score	Average Unweighted NASA-TLX
Plain Manipulation	**295.333**	38.792	39.063	47.500
Haptic Helpers	592.917	**19.167**	**34.688**	**46.927**

Table 5.9: Averages over all the participants' times, pressure changes, error score, and NASA-TLX Scores for each mode of drawing through telemanipulation tested. Using the haptic helpers reduces the changes in pressure while using the restraining helpers, decreases slightly the error score, and is mostly perceived as helping the user achieve better results. Although the helpers add an overhead in the time of completion, they helped the user attain better results.

Mobile Manipulation Task	Average Time $[m:s]$
Large Displacement Pick and Place	25:28
Opening a room's door	13:33

Table 5.10: Averages of the times taken to perform the large displacement pick and place towers of Hanoi and Opening a Door and exit a room cases.

extension of the setting used in the first three experiments. The overall result was that the mobile manipulator was capable of dealing with the unstructured environment. The second case study was performing an activity which, up to date, is an open problem: opening a door. Several robots, teleoperated and autonomous, have problems opening doors because of the different features doors may have and also because of the different challenges the task offers (unstructured environment, unknown interaction forces needed, geometric constraints, manipulability, etc.). The teleoperated mobile manipulator was capable to open the door in repeated times. The task was accomplished without failures of high-force interactions or controller instabilities. Table 5.10 shows the average of the times recorded for both case studies.

Chapter 6

Conclusions

This dissertation presented the development of a mobile manipulation test bed and the experiments that were conducted to test some of its features. The design goals were achieved while gaining insight on future directions of research using this system. In the following sections we summarize the contributions of this thesis (section 6.1) and finally we present some of the future topics which can expand this work (section 6.2).

6.1 Summary of Contributions

The system was programmed with a setting intended for stripped down teleoperation assuming no delay. This setting obviated the problems inherent to time delay to explore other aspects of a teleoperation system. Studies on which cameras help the operator manipulate more efficiently, alternative motion coordination schemes other than the classic clutching approach, and how different helpers can aid operators to improve their performance were investigated.

The available video feeds from a mobile manipulator usually are limited due to the available bandwidth. Assuming an unlimited bandwidth, we tested how providing different video feeds to the operators changed their performance. Most of the operators performed their best times with a static human inspired camera pair configuration overseeing the manipulation scene. These cameras provided the best times for most users. Despite this result, participants became unsure sometimes if they were doing the correct operations as this setting didn't provide complete information of the scene. All users stated that the eye in hand configuration was difficult due to the lack of depth perception and because the moving camera distracted them. This latter became evident also when giving all views to the operators. Most users also commented that although the eye in hand became distracting at times, it also became in handy when trying improve the accuracy of the placements.

The user studies also included an evaluation of different motion coordination schemes used to command the robots position when using an asymmetric system. The master, being almost ten times smaller in size and power than the slave, represented an interesting challenge in terms of which controller to use. We evaluated three different schemes. Clutching is the most generic and still the most well accepted by all users, although it introduces repetitive motions for large movements. Differential-End-Zone proved to be also well accepted by some users, while some others preferred the Position/Rate Switch. This latter was, at times, the most unstable and difficult to operate for some others. Despite their performance, these controller schemes might be good alternatives to the clutching approach.

Some other user studies were conducted to evaluate manipulation and haptic helpers. These helpers aided users by recording specific configurations of the slave and replaying them, or by restricting motions or the available workspace. The manipulation helpers proved to aid in most cases by relieving the users from doing several large motions, keeping straight paths, or maintaining a more even pressure over a surface than when doing the same task with plain teleoperation. Software helper routines decreased the perceived task load for the operators. In particular Manipulation Helpers reduced the completion times for the towers of Hanoi activity by almost half. Haptic Helpers aided users to keep more steady contact interactions when performing the fine manipulation task.

Case studies were also conducted to demonstrate the capabilities of the mobile manipulation setting. First, a large displacement pick and place activity was performed and the times, recorded. The robot successfully finished the task commanded solely using a Phantom Omni and a computer keyboard. The second case study was to open a door. In this case, the system also proved to be successful while rendering back the contact forces on the master device. This helped the operator to open the door and have the robot exit a room without major problems.

The overall system proved to be modular, expandable, and flexible. Two configurations using one and two arms were built. New components (either hardware or software) can be integrated with ease as long as the component can coexist with the others under the defined Framework.

The use of more elaborate software interfaces and message passing frameworks, can be both advantageous and taxing. While the software might have good support and resources, it might not fully be suited or intended for the application or it might require a lot of time invested into learning the basics of it. In this particular case, ROS was a very advantageous tool. The overall improvement in modularity alone out-weights the moderate

learning overhead. Some other advantages are the large community using this framework and the constant release of updates.

The use of off-the-shelf-components can largely speed-up the development process of a mobile manipulator. While *Ad hoc.* systems provide well integrated and design specific qualities, they also require a large share of resources. By using commercially available solutions, it is possible to achieve acceptable performance while keeping developing times shorter. Since the resources are more accessible, a larger community of researchers will be able to produce results that could speed up the advances in teleoperation, autonomy, and robot design, to name a few areas.

The methodology for designing a mobile manipulator presented in chapter 3 and the user interface rationale presented in chapter 4 were put to test by developing the mobile manipulator used to carry out the user and case studies presented in chapter 5. The system proved to be a stable test bed for numerous trials while also providing agile prototyping in hardware and software.

The developed system was designed to be capable of doing human-scale manipulation and prototyping test-bed for mobile manipulation algorithms. The design was largely thought to be modular. Modules can be interchanged, substituted, or added depending on what the system is going to be used for.

6.2 Future Work

Further refinement on the motion coordination schemes needs to be done. Specially on the Differential End Zone scheme. This controller was perceived as a very natural and intuitive scheme compared to clutching because it removed the onus of having to clutch several times to translate the mapped workspace of the arm to a new position. Some people suggested to include an option to disable the differential-end-zone.

The helpers proved to be useful and new helpers should be developed and tested. Extending the line primitive to work with general parametric paths and the plane primitive to work with parameterized surface are natural extensions. Moreover, helpers for mobile navigation could be another good extension. For the latter, it is first needed to do some more automation of the mobile base before the helpers can be added.

The design methodology presented is to be used as a guideline to develop mobile manipulators, yet still generic, addresses most of the common problems a designer is likely to find when sketching such a complex system. As a guideline, it is by no means exhaustive

and therefore should be combined with other design methodologies or frameworks. These combination and inclusions enrich the design process.

While ROS is a very good tool for prototyping and building stand-alone modules for robotics in general, it lacks the implementation of UDP package relay. This makes the system's overall latency prone to failures due to package retransmission when the package in question is already obsolete and useless. A UDP implementation of message passing can help the system work with the most up-to-date information at any given time.

In the same line of thought of the bandwidth and message passing, we observed that sometimes the system can become unresponsive for short periods of time (less than $100ms$) when commands are issued more frequently. This was always the case when using separate computers for the operator and the robot while using the video feeds from the cameras on-board. If the cameras were not used, or the on-board computer was also used as the operator's station, this problem was not observed. In future versions of the system, a detailed system characterization of the inter-module communication timing has to be constructed. This characterization will help solve this issue. As a temporary solution, it is possible to simply use a separate network for relaying the video back to the operators computer.

Although one of the intentions was to also study two arm manipulation and two-handed mobile manipulation, this avenue was only lightly explored and some results were presented in [40]. One-arm mobile manipulation was explored and the results were presented in this work yet, several further tests and different applications remain to be tried using the current setting.

Inclusion of sensors for the mobile navigation and path-planning should be tested and integrated into the system. The same way there are manipulation helpers, some navigation helpers can be developed to record particular positions of the mobile base in an environment.

Another application which remains to be addressed is the development of a controller scheme which considers the whole mobile manipulator as a single kinematic chain. So far, all the schemes presented in this work assume that the mobile manipulator is commanded in a segmented way, either commanding the base or commanding the manipulator, yet they can be extended to cope with this paradigm taking the robot as a single kinematic chain.

We believe that the inclusion of an active vision system can help operators to cope with the issues of the human inspired camera pair not giving complete information of the remote scene. Usually when doing an activity, humans tend to move their heads to find a better point of view that helps to solve the task. In this same way, an active vision system

comprised of a turret (pan and tilt) can help position the cameras and get better visual information of the remote scene.

The user studies conducted on how different camera views affected the performance of the operators provided good insight into how to train operators to use the information presented to them. It also provided insight on how to plan and include sensors when running on a tight budget that does not allow to use several camera feeds. Further studies should be made to evaluate if combinations of one of the cameras of the human inspired camera pair together with eye-in-hand are useful.

More sensors can be included into the system. Laser Rangers could help the mobile base not to collide with the environment while moving around. They could also serving as depth sensors for the arm and relay information and warn the operator of potential collisions. More cameras can also be included to gain more visual information of the surroundings of the slave. The information can be useful to make the operator more situational aware.

However useful the Phantom Omni proved to be, new haptic and non-haptic interfaces should be tested and interfaced to the system. The author used a haptics enabled game-pad controller to command the arm. Although the haptic feedback was not easy to decode as when using forces, it provided a condensed station to command the whole mobile manipulator.

Many other options for future work are possible, yet the ones above, from the point of view of the author, are the most important to develop in short. They provide the next building blocks to bring supervisory control for mobile manipulation closer.

Bibliography

[1] ActivMediaRobotics, "Manipulators," September 2008, brochure from ActiveMedia robotics. [Online]. Available: http://www.mobilerobots.com/Mobile_Robots.aspx

[2] N. Aeronautics and S. Administration. (2011, February) Robonaut mission to the international space station. internet. NASA. United States of America. [Online]. Available: http://robonaut.jsc.nasa.gov/iss/#issmission

[3] A. Albers, S. Brudniok, J. Ottnad, C. Sauter, and K. Sedchaicharn, "Upper body of a new humanoid robot - the design of armar iii," in *Proc. 6th IEEE-RAS Int Humanoid Robots Conf*, 2006, pp. 308–313.

[4] A. Albers and J. Ottnad, "Integrated structural and controller optimization for lightweight robot design," in *Proc. 9th IEEE-RAS Int. Conf. Humanoid Robots Humanoids 2009*, 2009, pp. 93–98.

[5] A. Albers, J. Ottnad, and C. Sander, "Development of a new wrist for the next generation of the humanoid robot armar," in *Proc. 8th IEEE-RAS Int. Conf. Humanoid Robots Humanoids 2008*, 2008, pp. 46–53.

[6] A. Albers, C. Sander, and A. Simsek, "Development of the actuation of a new wrist for the next generation of the humanoid robot armar," in *Proc. 10th IEEE-RAS Int Humanoid Robots (Humanoids) Conf*, 2010, pp. 677–682.

[7] A. Albers, S. Brudniok, J. Ottnad, C. Sauter, and K. Sedchaicharn, "Armar iii design of the upper body," Institute of Product Development University of Karlsruhe (TH), Tech. Rep., 2006. [Online]. Available: http://www.ldv.ei.tum.de/hcrs06/Assets/20.pdf

[8] R. O. Ambrose, R. T. Savely, S. M. Goza, P. Strawser, M. A. Diftler, I. Spain, and N. Radford, "Mobile manipulation using nasa's robonaut," in *Proc. IEEE Int. Conf. Robotics and Automation ICRA '04*, vol. 2, 2004, pp. 2104–2109.

[9] N. Ando, J.-H. Lee, and H. Hashimoto, "A study on influence of time delay in teleoperation," in *Proc. IEEE/ASME Int Advanced Intelligent Mechatronics Conf*, 1999, pp. 317–322.

[10] ——, "A study on influence of time delay in teleoperation - quantitative evaluation on time perception and operability of human operator," in *Proc. IEEE Int Systems, Man, and Cybernetics Conf. IEEE SMC '99*, vol. 5, 1999, pp. 1111–1116.

[11] T. Asfour, K. Welke, P. Azad, A. Ude, and R. Dillmann, "The karlsruhe humanoid head," in *Proc. 8th IEEE-RAS Int. Conf. Humanoid Robots Humanoids 2008*, 2008, pp. 447–453.

[12] T. Asfour, K. Berns, and R. Dillmann, "The humanoid robot armar: Design and control," in *Humanoids*, 2000, p. 6. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.108.7395&rep=rep1&type=pdf

[13] A. Aziminejad, M. Tavakoli, R. V. Patel, and M. Moallem, "Transparent time-delayed bilateral teleoperation using wave variables," vol. 16, no. 3, pp. 548–555, 2008.

[14] S. Bensalem, M. Gallien, F. Ingrand, I. Kahloul, and N. Thanh-Hung, "Designing autonomous robots," *IEEE Robotics & Automation Magazine*, vol. 16, no. 1, pp. 67–77, 2009.

[15] W. Bluethmann, R. Ambrose, M. Diftler, E. Huber, A. Fagg, M. Rosenstein, R. Platt, R. Grupen, C. Breazeal, A. Brooks, A. Lockerd, R. A. Peters, O. C. Jenkins, M. Mataric, and M. Bugajska, "Building an autonomous humanoid tool user," in *Proc. 4th IEEE/RAS Int Humanoid Robots Conf*, vol. 1, 2004, pp. 402–421.

[16] W. Bluethmann, R. . Ambrose, M. A. Difiler, S. Askew, E. Huber, S. M. Goza, F. Rehnmark, C. Lovchik, and D. Magruder, "Robonaut: A robot designed to work with humans in space," *Autonomous Robots*, vol. 14, pp. 179–197, 2003.

[17] C. Borst, C. Ott, T. Wimbock, B. Brunner, F. Zacharias, B. Bauml, U. Hillenbrand, S. Haddadin, A. Albu-Schaffer, and G. Hirzinger, "A humanoid upper body system for two-handed manipulation," in *Proc. IEEE Int Robotics and Automation Conf*, 2007, pp. 2766–2767.

[18] C. Borst, T. Wimbock, F. Schmidt, M. Fuchs, B. Brunner, F. Zacharias, P. R. Giordano, R. Konietschke, W. Sepp, S. Fuchs, C. Rink, A. Albu-Schaffer, and G. Hirzinger, "Rollin' justin - mobile platform with variable base," in *Proc. IEEE Int. Conf. Robotics and Automation ICRA '09*, 2009, pp. 1597–1598.

[19] G. Bradski, "The opencv library," *Dr. Dobb's Journal of Software Tools*, 2000.

[20] K. Brady and T.-J. Tarn, "Internet based manufacturing technology: intelligent remote teleoperation," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2000)*, vol. 2, 2000, pp. 843–848.

[21] R. Brooks, L. Aryananda, A. Edsinger, P. Fitzpatrick, C. Kemp, U. M. O'Reilly, E. Torres-Jara, P. Varshavskaya, and J. Weber, "Sensing and manipulating built-for-human environments," *International Journal of Humanoid Robotics*, vol. 1, no. 1, pp. 1–28, 2004.

[22] D. Brugali and P. Scandurra, "Component-based robotic engineering (part i) [tutorial]," *IEEE Robotics & Automation Magazine*, vol. 16, no. 4, pp. 84–96, 2009.

[23] D. Brugali and A. Shakhimardanov, "Component-based robotic engineering (part ii)," *IEEE Robotics & Automation Magazine*, vol. 17, no. 1, pp. 100–112, 2010.

[24] M. Buss, A. Peer, T. Schauss, N. Stefanov, U. Unterhinninghofen, S. Behrendt, G. Farber, J. Leupold, K. Diepold, F. Keyrouz, M. Sarkis, P. Hinterseer, E. Steinbach, B. Farber, and H. Pongrac, "Multi-modal multi-user telepresence and teleaction system," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems IROS 2008*, 2008, pp. 4137–4138.

[25] M. Buss, K.-K. Lee, N. Nitzsche, A. Peer, B. Stanczyk, and U. Unterhinninghofen, *Advanced Telerobotics: Dual-Handed and Mobile Remote Manipulation*. Springer Berlin / Heidelberg, 2007, ch. 28, pp. 471–497.

[26] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementation*, ser. Intelligent Robotics and Autonomous Agents, R. C. Arkin, Ed. Cambridge, Massachussetts, USA: The MIT Press, 2005.

[27] G. Christiansson, "Wave variables and the 4 channel architecture for haptic teleoperation," in *Haptics: Perception, Devices and Scenarios*, ser. Lecture Notes in Computer Science, M. Ferre, Ed. Springer Berlin / Heidelberg, 2008, vol. 5024, pp. 169–174.

[28] M. O. Culjat, C.-H. King, M. L. Franco, C. E. Lewis, J. W. Bisley, E. P. Dutson, and W. S. Grundfest, "A tactile feedback system for robotic surgery," in *Proc. 30th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society EMBS 2008*, 2008, pp. 1930–1934.

[29] P. Deegan, R. Grupen, A. Hanson, E. Horrell, S. Ou, E. Riseman, S. Sen, B. Thibodeau, A. Williams, and D. Xie, "Mobile manipulators for assisted living in residential settings," *Autonomous Robots, Special Issue on Socially Assistive Robotics*, vol. 24, no. 2, p. 14, 2008.

[30] P. Deegan, B. J. Thibodeau, and R. Grupen, "Designing a self-stabilizing robot for dynamic mobile manipulation," in *RSS*, 2006. [Online]. Available: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA459932&Location=U2&doc=GetTRDoc.pdf

[31] F. Devernay, F. Mourgues, and E. Coste-Maniere, "Towards endoscopic augmented reality for robotically assisted minimally invasive cardiac surgery," in *Proc. Int Medical Imaging and Augmented Reality Workshop*, 2001, pp. 16–20.

[32] R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, Carnegie Mellon University, Robotics Institute, August 2010. [Online]. Available: http://www.programmingvision.com/rosen_diankov_thesis.pdf

[33] M. A. Diftler, C. J. Culbert, R. O. Ambrose, J. Platt, R., and W. J. Bluethmann, "Evolution of the nasa/darpa robonaut control system," in *Proc. IEEE Int. Conf. Robotics and Automation ICRA '03*, vol. 2, 2003, pp. 2543–2548.

[34] W. Ding, L. Pei, H. Li, N. Xi, and Y. Wang, "The effects of time delay of internet on characteristics of human behaviors," in *Proc. Int. Conf. Networking, Sensing and Control ICNSC '09*, 2009, pp. 502–506.

[35] EUROP Secretariat EUnited Robotics. (2011, December) eurobotics challenge 2011. European Robotics Association. [Online]. Available: http://www.eurobotics-project.eu/eurobotics-challenge/eurobotics-challenge.html

[36] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. Parallel virtual machine. [Online]. Available: http://www.csm.ornl.gov/pvm/

[37] A. Goto, R. Inoue, T. Tezuka, and H. Yoshikawa, "A research on tele-operation using virtual reality," in *Proc. Workshop th IEEE Int Robot and Human Communication RO-MAN'95 TOKYO*, 1995, pp. 147–152.

[38] G. Hager, S. Puri, K. Toyama, and D. Burschka, "A brief tour of xvision," —http://www.cs.jhu.edu/CIPS/xvision/index.html, 2010. [Online]. Available: http://www.cs.jhu.edu/CIPS/xvision/index.html

[39] S. G. Hart and L. E. Stavenland, "Development of NASA-TLX (Task Load Index): results of empirical and theoretical research," in *Human Mental Workload*, P. A. Hancock and N. Meshkati, Eds. Elsevier, 1988, ch. 7, pp. 139–183. [Online]. Available: http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20000004342_1999205624.pdf

[40] A. Hernandez-Herdocia, A. Shademan, and M. Jagersand, "Building a mobile manipulator from off-the-shelf components," in *Proc. IEEE/ASME Int Advanced Intelligent Mechatronics (AIM) Conf*, 2010, pp. 1116–1121.

[41] B. M. Horowitz and J. H. Lambert, "Assembling off-the-shelf components: "learn as you go" systems engineering," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 36, no. 2, pp. 286–297, 2006.

[42] D. Katz, E. Horrell, Y. Yang, B. Burns, T. Buckley, A. Grishkan, V. Zhylkovskyy, O. Brock, and E. L. Miller, "The umass mobile manipulator uman: An experimental platform for autonomous mobile manipulation," in *In Workshop on Manipulation in Human Environments, at Robotics: Science and Systems*, 2006. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.123.3876&rep=rep1&type=pdf

[43] N. Klopcar and J. Lenarcic, "Kinematic model for determination of human arm reachable workspace," *Meccanica*, vol. 40, no. 2, pp. 203–219, 2005.

[44] P. Kremer, T. Wimbock, J. Artigas, S. Schatzle, K. Johl, F. Schmidt, C. Preusche, and G. Hirzinger, "Multimodal telepresent control of dlr's rollin' justin," in *Proc. IEEE Int. Conf. Robotics and Automation ICRA '09*, 2009, pp. 1601–1602.

[45] S. Leonard. openman: An open source C++ toolbox for control and simulations of manipulators. [Online]. Available: http://sourceforge.net/projects/openman/

[46] G. Leroy, *Designing User Studies in Informatics*, ser. Health Informatics, K. J. Hannah and M. J. Ball, Eds. Springer, 2011.

[47] D. Lovi, N. Birkbeck, A. Hernandez-Herdocia, A. Rachmielowski, M. Jagersand, and D. Cobzas, "Predictive display for mobile manipulators in unknown environments using online vision-based monocular modeling and localization," in *Proc. IEEE/RSJ Int Intelligent Robots and Systems (IROS) Conf*, 2010, pp. 5792–5798.

[48] M. J. H. Lum, J. Rosen, H. King, D. C. W. Friedman, T. S. Lendvay, A. S. Wright, M. N. Sinanan, and B. Hannaford, "Teleoperation in surgical robotics – network latency effects on surgical performance," in *Proc. Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society EMBC 2009*, 2009, pp. 6860–6863.

[49] M. J. H. Lum, J. Rosen, T. S. Lendvay, A. S. Wright, M. N. Sinanan, and B. Hannaford, "Telerobotic fundamentals of laparoscopic surgery (fls): Effects of time delay - pilot study," in *Proc. 30th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society EMBS 2008*, 2008, pp. 5597–5600.

[50] E. Marchand, F. Spindler, and F. Chaumette, "ViSP for visual servoing: a generic software platform with a wide class of robot control skills," *IEEE Robotics & Automation Magazine*, vol. 12, no. 4, pp. 40–52, 2005.

[51] M. C. Martin, "Controlling cardea: Fast policy search in a high dimensional space," MIT Media Laboratory, Massachusetts Institute of Technology Cambridge, MA 02139, Vision and Modelling 583, June 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.153.6748&rep=rep1&type=pdf

[52] J. S. Mehling, P. Strawser, L. Bridgwater, W. K. Verdeyen, and R. Rovekamp, "Centaur: Nasa's mobile humanoid designed for field work," in *Proc. IEEE Int Robotics and Automation Conf*, 2007, pp. 2928–2933.

[53] T. Morita, H. Iwata, and S. Sugano, "Human symbiotic robot design based on division and unification of functional requirements," in *Proc. IEEE Int. Conf. Robotics and Automation ICRA '00*, vol. 3, 2000, pp. 2229–2234.

[54] B. K. Muirhead, "Mars rovers, past and future," in *Proc. IEEE Aerospace Conf*, vol. 1, 2004.

[55] C. Ott, O. Eiberger, W. Friedl, B. Bauml, U. Hillenbrand, C. Borst, A. Albu-Schaffer, B. Brunner, H. Hirschmuller, S. Kielhofer, R. Konietschke, M. Suppa, T. Wimbock, F. Zacharias, and G. Hirzinger, "A humanoid two-arm system for dexterous manipulation," in *Proc. 6th IEEE-RAS Int Humanoid Robots Conf*, 2006, pp. 276–283.

[56] C. Ott and Y. Nakamura, "Employing wave variables for coordinated control of robots with distributed control architecture," in *Proc. IEEE Int. Conf. Robotics and Automation ICRA 2008*, 2008, pp. 575–582.

[57] P. K. Pook and D. H. Ballard, "Deictic teleassistance," in *Proc. IEEE/RSJ/GI Int. Conf. Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World' IROS '94*, vol. 1, 1994, pp. 245–252.

[58] P. K. Pook, "Teleassistance: Using deictic gestures to control robot action," Ph.D. dissertation, University of Rochester, 1995. [Online]. Available: ftp://192.5.53.208/pub/papers/robotics/95.tr594thesis.Teleassistance_Deictic_gestures_control_robot_action.ps.gz

[59] G. Pour, "Moving toward component-based software development approach," in *Proc. Technology of Object-Oriented Languages TOOLS 27*, 1998, pp. 296–300.

[60] ——, "Towards component-based software engineering," in *Proc. Twenty-Second Annual Int. Computer Software and Applications Conf. COMPSAC '98*, 1998.

[61] C. Preeda, G. G. Hwang, and H. Hashimoto, "Vr simulator for nano smms teleoperation over the delayed networks," in *Proc. Int SICE-ICASE Joint Conf*, 2006, pp. 4826–4831.

[62] M. Quigley, S. Batra, S. Gould, E. Klingbeil, Q. Le, A. Wellman, and A. Y. Ng, "High-accuracy 3d sensing for mobile manipulation: Improving object detection and door opening," in *Proc. IEEE Int. Conf. Robotics and Automation ICRA '09*, 2009, pp. 2816–2822.

[63] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," in *Open-source software workshop*, IEEE-RAS. Kobe, Japan: IEEE Int. Conf. Robotics and Automation ICRA '09, May 2009. [Online]. Available: http://pub1.willowgarage.com/~konolige/cs225B/docs/quigley-icra2009-ros.pdf

[64] A. Rachmielowski, N. Birkbeck, M. Jagersand, and D. Cobzas, "Realtime visualization of monocular data for 3d reconstruction," in *Proc. Canadian Conf. Computer and Robot Vision CRV '08*, 2008, pp. 196–202.

[65] F. Rehnmark, W. Bluethmann, J. Mehling, R. O. Ambrose, M. Diftler, M. Chu, and R. Necessary, "Robonaut: the 'short list' of technology hurdles," *Computer*, vol. 38, no. 1, pp. 28–37, 2005.

[66] S. E. Salcudean, N. M. Wong, and R. L. Hollis, "Design and control of a force-reflecting teleoperation system with magnetically levitated master and wrist," vol. 11, no. 6, pp. 844–858, 1995.

[67] T. B. Sheridan, "Space teleoperation through time delay: review and prognosis," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 5, pp. 592–606, 1993.

[68] ——, *Telerobotics, Automation, and Human Supervisory Control*. The MIT Press, August 1992.

[69] ——, *Handbook of Human Factors and Ergonomics*, 3rd ed. Wiley, January 2006, handbook Supervisory Control, pp. 1025–1052.

[70] C. Smith and H. I. Christensen, "Using cots to construct a high performance robot arm," in *Proc. IEEE Int Robotics and Automation Conf*, 2007, pp. 4056–4063.

[71] S. Srinivasa, D. Ferguson, C. Helfrich, D. Berenson, A. Collet, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and M. VandeWeghe, "Herb: A home exploring robotic butler," *Autonomous Robots*, vol. 28, pp. 5 – 20, 2009.

[72] M. Stilman, J. Olson, and W. Gloss, "Golem krang: Dynamically stable humanoid robot for mobile manipulation," in *Proc. IEEE Int Robotics and Automation (ICRA) Conf*, 2010, pp. 3304–3309.

[73] D. K. Stosic, W. A. Hatch, J. P. Lux, and R. L. McMaster, "Utilizing off-the-shelf parts for the next generation of space exploration," in *Proc. Aerospace Conf. IEEE*, vol. 2, 2001.

[74] Sugano Laboratory, Waseda University, "Twendy-one," February 2008. [Online]. Available: http://twendyone.com/index_e.html

[75] M. Tavakoli, A. Aziminejad, R. V. Patel, and M. Moallem, "Stability of discrete-time bilateral teleoperation control," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems IROS 2007*, 2007, pp. 1624–1630.

[76] M. Tavakoli and R. D. Howe, "Improving teleoperation performance in the presence of non-ideal robot dynamics," in *Proc. IEEE Int. Conf. Technologies for Practical Robot Applications TePRA 2008*, 2008, pp. 128–130.

[77] B. J. Thibodeau, P. Deegan, and R. Grupen, "Static analysis of contact forces with a mobile manipulator," in *Proc. IEEE Int. Conf. Robotics and Automation ICRA 2006*, 2006, pp. 4007–4012.

[78] University of Massachussetts at Amherst, "Robotics at umass amherst - robots - dexter," http://www-robotics.cs.umass.edu/Robots/Dexter, September 2008, the information was taken from the UMASS at Amherst Lab of Perceptual Robotics Webpage. [Online]. Available: http://www-robotics.cs.umass.edu/Robots/Dexter

[79] S. V. Velanas and C. S. Tzafestas, "Human telehaptic perception of stiffness using an adaptive impedance reflection bilateral teleoperation control scheme," in *Proc. IEEE RO-MAN*, 2010, pp. 21–26.

[80] L. E. P. Williams, R. B. Loftin, H. A. Aldridge, E. L. Leiss, and W. J. Bluethmann, "Kinesthetic and visual force display for telerobotics," in *Proc. IEEE Int. Conf. Robotics and Automation ICRA '02*, vol. 2, 2002, pp. 1249–1254.

[81] Willow-Garage. (2010, January) Technical specifications. [Online]. Available: http://www.willowgarage.com/pages/robots/technical-specs

[82] M. C. Yip, M. Tavakoli, and R. D. Howe, "Performance analysis of a manipulation task in time-delayed teleoperation," in *Proc. IEEE/RSJ Int Intelligent Robots and Systems (IROS) Conf*, 2010, pp. 5270–5275.

[83] M. Zhu, Y. Ge, S. Huang, and Y. Jiang, "Experimental analysis of communication quality of teleoperation with force feedback via ip network," in *Proc. IEEE Int. Conf. Integration Technology ICIT '07*, 2007, pp. 155–160.

Appendix A

Experimental Design

A.1 The Towers of Hanoi

The towers of Hanoi is a mathematical puzzle consisting of a number of places and a number of discs of different diameters. The puzzle starts with the disks ordered in ascending size in one place: the largest at the bottom and the smallest at the top. The objective of the game is to move the entire stack from one place to another by moving the top-most disk in the stack, one at a time, without putting a bigger disk on top of a smaller one.

The towers of Hanoi was selected as a proxy for general teleoperation settings where pick and place are the principal actions involved. In [35] this setting was used for their autonomous mobile manipulation challenge because it gives an excellent sandbox problem to address research topics like perception, world modeling, planning, and coordination. Although the focus is in teleoperation and not directly in autonomy, it is possible to use the same setting to experiment with **Operator Interfaces**, **Functionalities**, and **System Integration**. This same setting can later be used to experiment as well with **Control Stability**, **Time-Delay**, and **Safety**.

For the experiments, a three-place three-disk instance of the towers of Hanoi was used. Since the focus of the study is not the solution of the puzzle, the participants were given the strategy to solve the problem in advance. The strategy is presented in procedure 7.

There are six different configurations that can be used for the trials:

- left stack to center stack
- left stack to right stack
- center stack to left stack
- center stack to right stack

Procedure 7 Strategy to Solve the Towers of Hanoi with 3 disks and 3 places

1: Move the smallest disk to the “target” stack
2: Move the medium disk to the remaining stack
3: Move the smallest disk on top of the medium disk
4: Move the biggest disk to the “target” stack
5: Move the smallest disk to the “starting” stack
6: Move the medium disk on top of biggest disk
7: Move the smallest disk on top of the medium disk

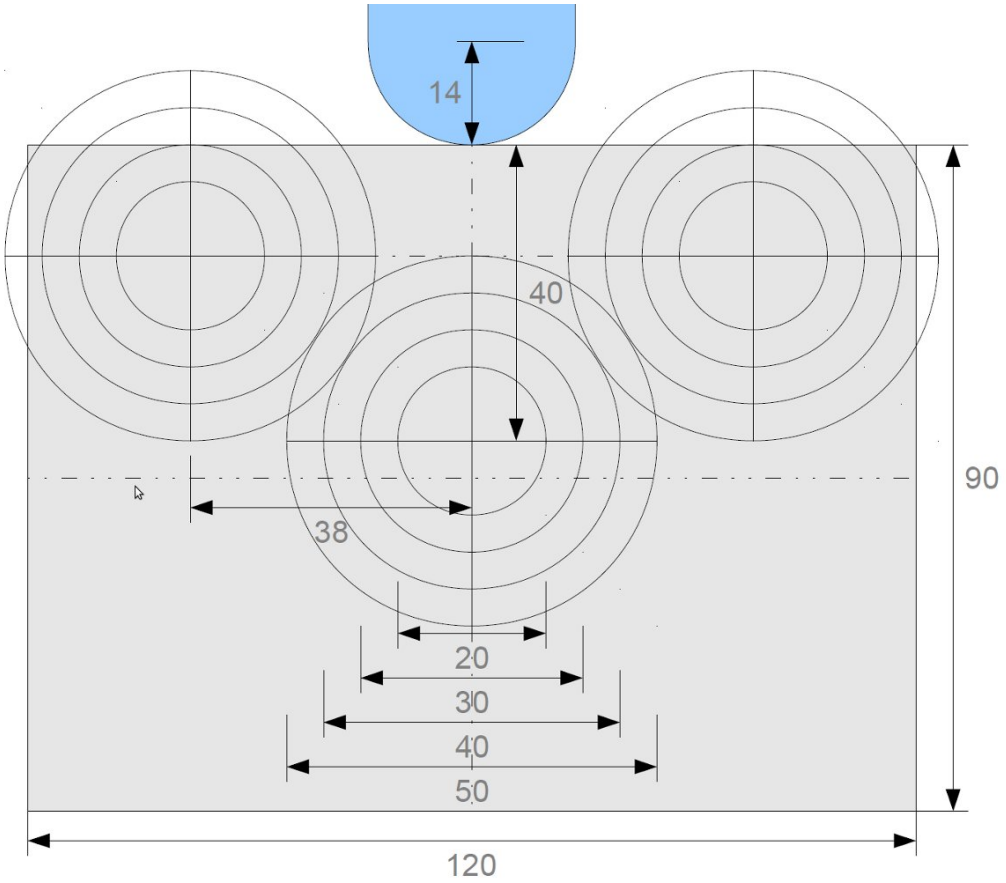


Figure A.1: An annotated diagram of the arrangement of the Towers of Hanoi setting. The dimensions are in centimeters.

- right stack to left stack
- right stack to center stack

The stacks were marked with two colored targets which were placed on top of a black surface. The places were 50*cm* in diameter and were placed approximately 45*cm* apart one from the other. The center stack was placed 54*cm* away from center of the arm’s base directly in front of the arm. An annotated diagram is presented in figure A.1 and

The disks used in the experiments were cylindrical of about 5*cm* high and with diameters of 7*cm*, 10*cm*, and 13*cm*. They were made from plastic foam and covered with a black and white pattern to allow for the center of the piece to be found even when occluded. This feature was also used in the design of the targets for the stacks/places.

The target-like setting served two purposes. The first was to point out the exact location where the users were required to place the discs and build the stacks. The second is to be able to measure the accuracy of the placement. For the sake of simplicity this accuracy was discretized into four possible zones. The first circle was selected to be 20*cm* in diameter so it could allow the biggest disk to be placed with some degree of error. The subsequent circles were defined at 30*cm*, 40*cm*, and 50*cm*. Users were advised to place the discs as close as possible to the center and within the smallest circle.

The test-bed was left open in the edges and a disk can leave the table at any time if mishandled. If this should happen, the trial will be restarted. The purpose of this feature is to differentiate from two different failures or errors during manipulation: recoverable and fatal errors. The former was usually the case when pieces turned over their side while on the table. The later happened when one disk left the testbed. These two types of errors are always present in manipulation settings[1].

A.2 Drawing

Drawing (*i.e.,* making traces with a stylus) is a very basic activity which requires precision movements and the use of a tool which deposits a material on top of a surface along a path. Drawing can be used as a proxy for any kind of application which requires fine manipulation constrained to a plane or along a path. Examples of such activities are welding, glue application, painting, and inspection.

In this case the main task is to follow some patterns which include straight lines, circular segments, or a combination of both. These "basic" configurations were selected because they can be extended in future iterations to general curve segments and non-planar paths. Figure A.2 shows the guide pattern the users were asked to follow and draw over.

The drawing has some specific dimensions. The P was selected to be 20*cm* high and 12*cm* long while the triangle was 10*cm* base by 10*cm* high and the circle 10*cm* diameter. The shaded region is 1*cm* thick and is the region where the user was asked to trace the lines or curves. The patterns were printed in a letter-size sheet of paper with a landscape

[1] some cases, like in telesurgery or in space teleoperation, the recoverable errors are very few, and most errors can be indeed fatal

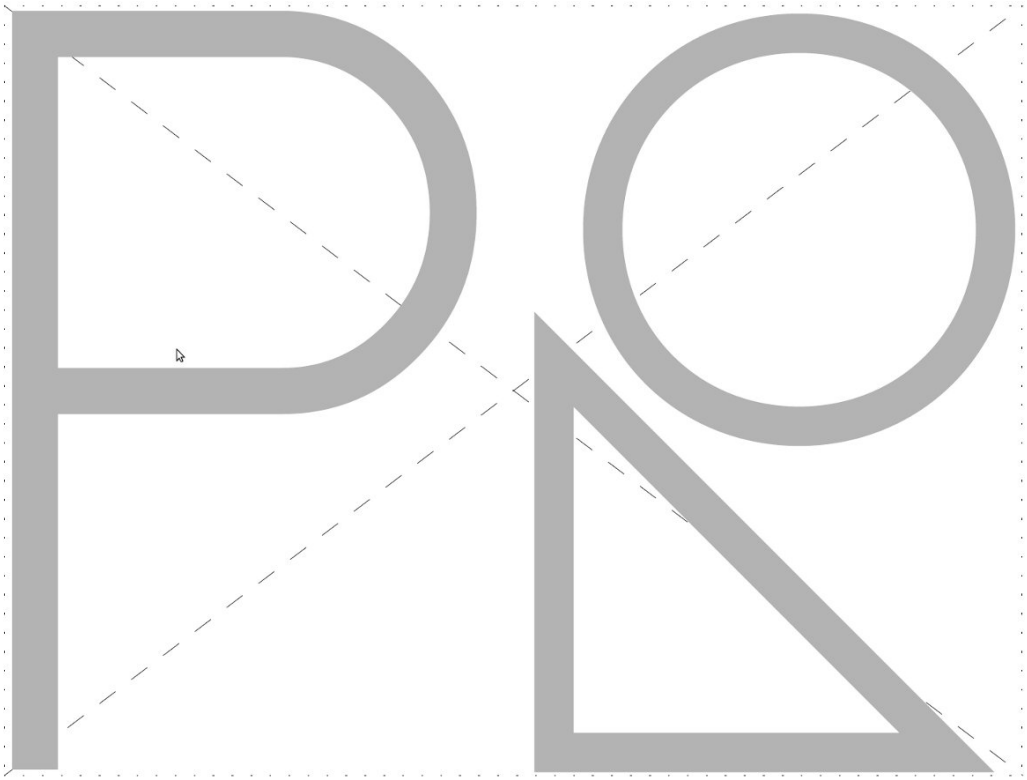


Figure A.2: These are the patterns used in the drawing task. The “P” combines straight lines and curves, while the circle is a pure curve and the triangle has only straight lines.

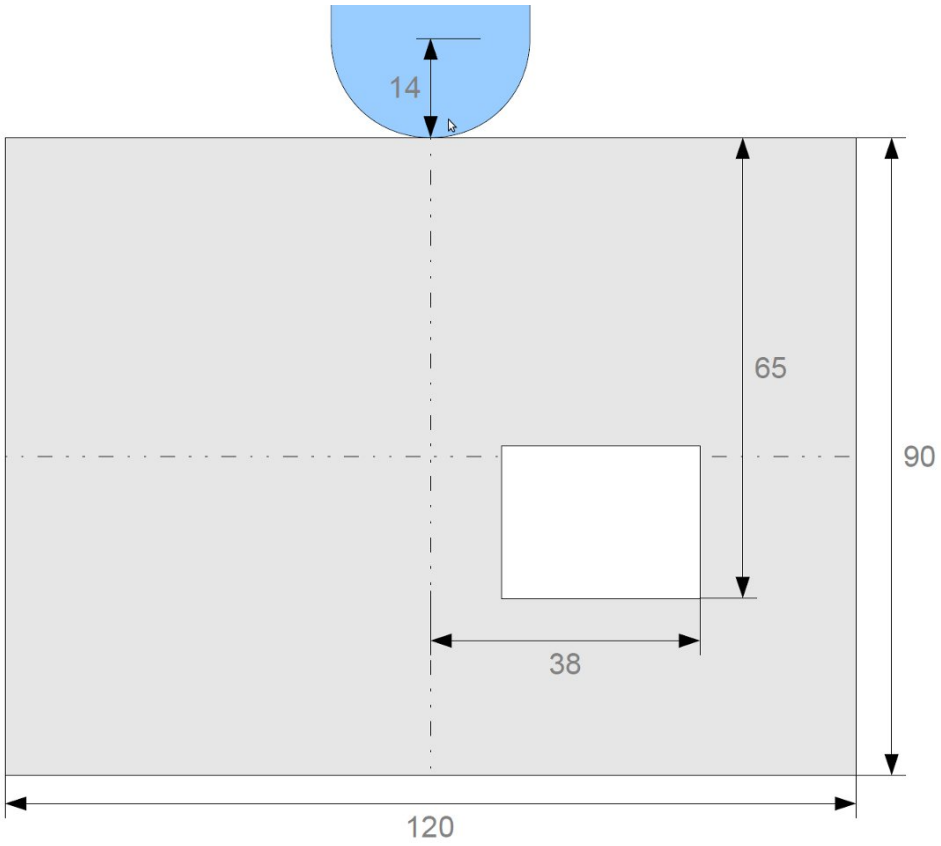


Figure A.3: An annotated diagram of the arrangement of the Drawing setting. The dimensions are in centimeters.

orientation. The upper left corner of the sheet of paper was placed 79*cm* away from the robot base's center and 38*cm* to the left of the centerline. This arrangement was selected so that the cameras had clear line of sight of the paper at all times without occlusions from the arm or the drawing tool. Figure A.3 shows an annotated diagram of the testbed.

A felt tip marker was used as the drawing stylus as it made it easy to evaluate how smooth the traces were done and differentiate levels of pressure applied by the marker to the paper. This can be seen after the trial has been completed by looking at the thickness of the trace and the amount of ink that penetrated into the paper *i.e.,* by looking at the paper from behind, this features become evident.

A way for the robot to hold a marker was needed. Although the gripper itself could hold the marker, a more "ergonomic" interface was favored. This interface served two purposes: Put the marking pen tip directly in front of the eye-in-hand camera and also put it away from the gripper so it would be easier to look at the point from the left camera of the overview pair. The second purpose was to be able to change quickly between the settings

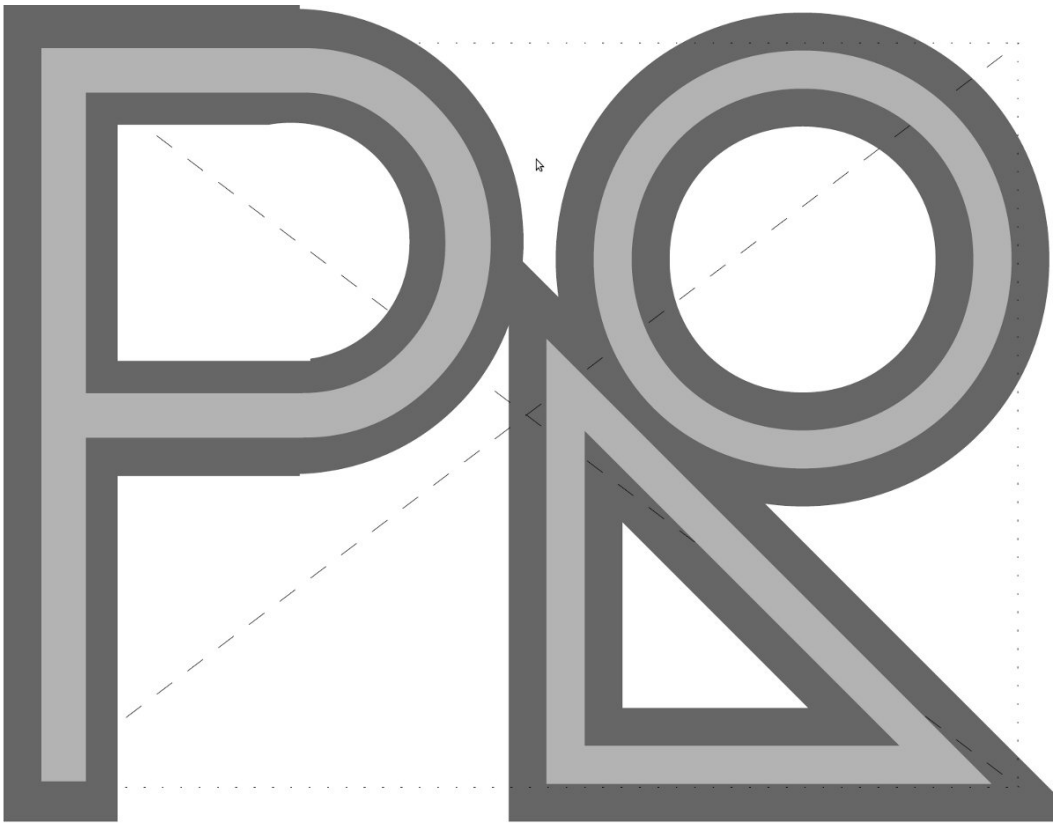


Figure A.4: The overlay screen used to identify and classify the different errors in the drawings.

of drawing and the towers of Hanoi while provide a mechanical "fuse" to prevent the arm from suffering damage due to contact with the table[2]. The contraption made it possible to adjust the angle of the pen with respect to the drawing plane as well.

To evaluate the drawings qualities, an overlay pattern was used defining three regions: safe, minor error, and too-far-error. The zones were selected to be:

Safe $< \pm 0.5cm$ from the original path

Minor error $> \pm 0.5cm$ and $< \pm 1.5cm$ from the original path

too-far-error $> \pm 1.5cm$ away from the path.

A detailed view of this overlay is shown in figure A.4. In this case it was decided to deem all errors recoverable and thus each trial was run until all traces were completed by the user. In a post stage, the pressure and the smoothness of the traces were evaluated by

[2]Although the table was made of cardboard and it would have collapsed if the arm had a collision with it.

looking at the completed drawings. Identification of the changes in thickness and intensity of the traces in the back of the sheet was done manually.

A.3 The Set of Experiments

For all the experiments, it was decided to run each condition three times in order to be able to gather enough data to minimally characterize a user. Each trial was expected to take between 2 and 6 minutes each. It was decided to make the minimum number of trials so the tests would not take longer than a couple of hours, otherwise variability due to tiredness or boredom would appear in the data. Overall the conditions to be tested per experiment were:

- 3 Camera Feeds Experiments (Towers of Hanoi)
- 3 Control Scheme Experiments (Towers of Hanoi)
- 2 Manipulation Helpers Experiments (Towers of Hanoi)
- 2 Haptic Helpers Experiments (Drawing)

An important fact about all the experiments involving the towers of Hanoi is that one of the conditions corresponds to a common configuration in all the experiments. This configuration has all three camera feeds, uses the clutching control scheme, no helpers are allowed and the mobile base is disabled. This common configuration made it possible to run this condition only once (three trials per user) and use it for every experiment. At the end of the day, this condition is the "control" of each experiment. Therefore, only 6 conditions need to be run for the towers of Hanoi and 2 conditions for the Drawing experiments:

- 3 Camera Feeds (control conditions included here)
- 2 Master-Slave Motion Coordination Schemes
- 1 Manipulation Helpers
- 2 Haptic Helpers

The Haptic Helper experiments was treated as a separate module amd it was run after the Towers of Hanoi experiments were completed. Training the users and let them become familiar with the system was desirable before requesting an activity requiring fine manipulation and system awareness.

A natural ordering of the experiments would be first to run the Camera Feeds, follow with the Manipulation Helpers, then the Master-Slave Motion Coordination Schemes and finally the Haptic Helpers. This ordering assumes that the user will train just enough to control the system. Since the camera video feeds are somewhat detached from the controller related experiments (Manipulation Helpers and Master-Slave Motion Coordination schemes), it is possible to start with this setting. This let the users gain confidence and skill commanding the robot. Once the users have completed the camera feed tests, they become comfortable with the setting and were taught to use the manipulation helpers. Finally, since the users are more familiar on how the system works and behaves, it is safe to introduce the novelty of using different motion coordination schemes to command the robot.

In the camera feeds experiments data for the 3 conditions was collected; among those, the data for the trials under "controlled-conditions" was also gathered. The trials corresponding to the manipulation helpers can be now completed and the motion coordination schemes' trials recorded. Trials for an experiment were randomized. This is explained in more detail in the next subsection (A.4).

Six conditions should be tested. If all trials took 6 minutes and all trials are run one behind the other, one set of trials would require each participant to use the system effectively for 36 minutes. Since one single run may not be sufficient to characterize the user, a natural small number is 3 trials per condition. The, a total time of 108 minutes is required per operator. If more trials were to be run, the span of time needed to complete the trials would increase significantly. Finding volunteers for the trials would be difficult. Moreover, given the mental and physical demand of a teleoperation setting, more than three hours of use would likely pollute the trials with the effects of tiredness from the users.

The 108 minutes do not include the training time, which could easily add up to 200 minutes. A desirable action then would be to separate the trials into two sessions of around 60 minutes each (only for the trials). In the first session it would be desirable to train the user and then go directly into the trials for the camera feeds. Then on a separate session give some time for a warm-up and training on how to use the manipulation helpers, run the trials for them, train the users with the different controller schemes, and finally run the trials. Both sessions would be of around 150 minutes.

A last session would be required to complete the drawing related experiments yet this session would be a lot less lengthy as the users are familiar with the system. A more detailed chronology of the experiments is presented in subsection A.4.1

A.4 Randomized Trials

One important consideration when designing experiments is to randomize the trials to avoid potential bias. The trials for all the experiments were randomized in the ordering in which the control conditions and the alternate conditions were presented to the participant. Only for those experiments using the towers of Hanoi, the initial/final configuration of the towers was randomized as well.

For the towers of Hanoi, in both cases (ordering and instance of the puzzle) a non-substituting approach was used; all users would face three, and exactly three, trials for each of the tested conditions and no two trials testing the same conditions would use the same instance of the puzzle.

All instances of the puzzle have exactly 7 movements according to the strategy which was given to the participants. This seven manipulations, have another 7 corresponding motions of alignment (without holding a piece) giving a total of 14 movements. Each of these movements can be classified as a long (between left and right stacks) or short (between left and center or right and center) motions. The instance of the puzzle does not add variability from the point of view of the test, yet helps to keep interest of the participant in the task.

From the three experiments involving the towers of Hanoi, only two were randomized. The experiment where the manipulation helpers are used, could not be randomized in the ordering in which the control conditions and the testing conditions are presented to the user. This was because only one of such alternate conditions was left to be recorded since the control conditions have been recorded already. The instance of the puzzle was still randomized.

For the case of the drawing experiments, the ordering of the conditions presented to the user was randomized. This random ordering was also done by non-substitution and therefore all subjects will face exactly three trials for each of the conditions (Haptic Helpers or Plain teleoperation).

The ordering and the conditions were communicated to the users at the beginning of each trial as they should know which tools they are allowed to use to complete the task.

A.4.1 A Participant's Schedule

All eight participants in the trials were asked to schedule their available times to perform the tests. They were asked to schedule one session at a time and then schedule the next session

when the previous one had been completed. The full schedule of a participant would look as follows:

Session 1

General Training *[5 to 60 minutes]* Explanation of the generalities of the system and instruction on how to operate the arm. The objective is to get familiarized with the setting of the display showing the video feeds and the way to command the arm using the Phantom Omni. Training on clutching control scheme, operate the gripper and disengage the control station from the robot. Practice goes on until user feels comfortable with the setting

Camera Feed Training *[15 to 60 minutes]* Practice solving some instances (up to two per video feed mode) of the towers of Hanoi puzzle, first with all three cameras, then with the overview camera pair, and finally with the eye-in-hand. All users are required to test all three video feed modes.

Optional Rest *[Up to 20 minutes]*

Camera Feed Trials *[40 to 90 minutes]* The user runs the 9 trials (3 with all 3 video feeds, 3 with overview camera pair, and 3 with eye-in-hand). Optional 5 minute breaks between trials can be taken at will by the participants

The user can start the next session right away, but it is suggested for the user to take at least a 24 hour break before the next session.

Session 2

Warm-up *[5 to 15 minutes]* review the operation of the system by solving one instance of the towers of Hanoi.

Place Helpers Training *[30 to 60 minutes]* Explanation and familiarization with the use of recording and replaying places in the workspace of the robot. Practice with solving up to three instances of the towers of Hanoi.

Optional Rest *[Up to 20 minutes]*

Place Helpers Trials *[15 to 45 minutes]* The user runs 3 trials using place helpers to solve the towers of Hanoi.

Optional Rest *[Up to 20 minutes]*

Control Scheme Training: DEZ *[10 to 25 minutes]* The user receives an explanation of the Differential End Zone control scheme. User practices first in the void workspace of the arm to get familiarized with the behavior of the control scheme and then solves up to 3 instances of the towers of Hanoi puzzle.

Control Scheme Training: P/R Switch *[5 to 15 minutes]* The user receives the explanation of the Position/Rate Switch control scheme . The user again practices in the void workspace of the arm to familiarize with the behavior of the scheme and then solves up to 3 instances of the towers of Hanoi puzzle.

Optional Rest *[Up to 20 minutes]*

Control Scheme Trials *[20 to 60 minutes]* The user runs 6 trials (3 with Differential End Zone and 3 with Position/Rate Switch). Optional 5 minute breaks between trials can be taken at will by the participants

The user can start the next session right away, but it is suggested for the user to take at least a 24 hour break before the next session.

Session 3

Warm-up *[5 minutes]* review the operation of the system by servoing to the drawing area. Users will be notified about the differences in the camera video feeds and about the difference in the scaling of the control commands.

Drawing Training *[5 to 15 minutes]* Users will be asked to draw curves and lines on a white letter-sized paper using the robot. The drawing at this point is considered "freehand" as no other helpers other than the haptic feedback from the system is present.

Line and Plane Haptic Helpers Training *[10 to 45 minutes]* Users get explanation of the use and particularities of the haptic helpers to constrain the robot to remain in a plane or within a line. Users are then asked to draw lines and curves using these two haptic helpers.

Optional Rest *[Up to 20 minutes]*

Setting	Value
Video Feed	All three camera feeds.
Arm Control Scheme	Clutching (Variable Scaling), Differential End Zone and P/R Switch; Changed at will by operator
Helpers	Manipulation and Haptic Helpers available, Zero, Control Disengage, Switch to joint-space control
Mobile Base	Enabled with Rate Control
Activity	Towers of Hanoi and Opening a Door

Table A.1: The configuration used in the mobile manipulation case studies.

Drawing Trials *[25 to 60 minutes]* Users draw over the presented patterns 6 times (one time per trial, 3 with haptic helpers and 3 freehand).Optional 5 minute breaks between trials can be taken at will by the participants.

A.5 Mobile Manipulation Case Studies

In both case studies the robot had several options enabled to be changed on-the-fly. The overall setting is summarized in table A.1.

A.5.1 Revisiting the Towers of Hanoi

A detailed floor-plan of the setting of the experiment is shown in figure A.5.

Unlike the setting used in the experiments, where the stacking places were at an even height of the manipulator's base, all three stacking places in this setting had different heights, yet all the stacking places had an effective area of 40cm by 40cm. The stacks had the following heights:

- Place on the right 76cm off the ground
- Place on the low-left 62cm off the ground
- Place on the top-left 68cm off the ground

The trials used and the times taken for those were:

- From low-left to top-left; done in 24 $minutes$, 58 $seconds$.
- From top-left to right; done in 25 $minutes$, 32 $seconds$.
- From right to low-left; done in 25 $minutes$, 53 $seconds$.

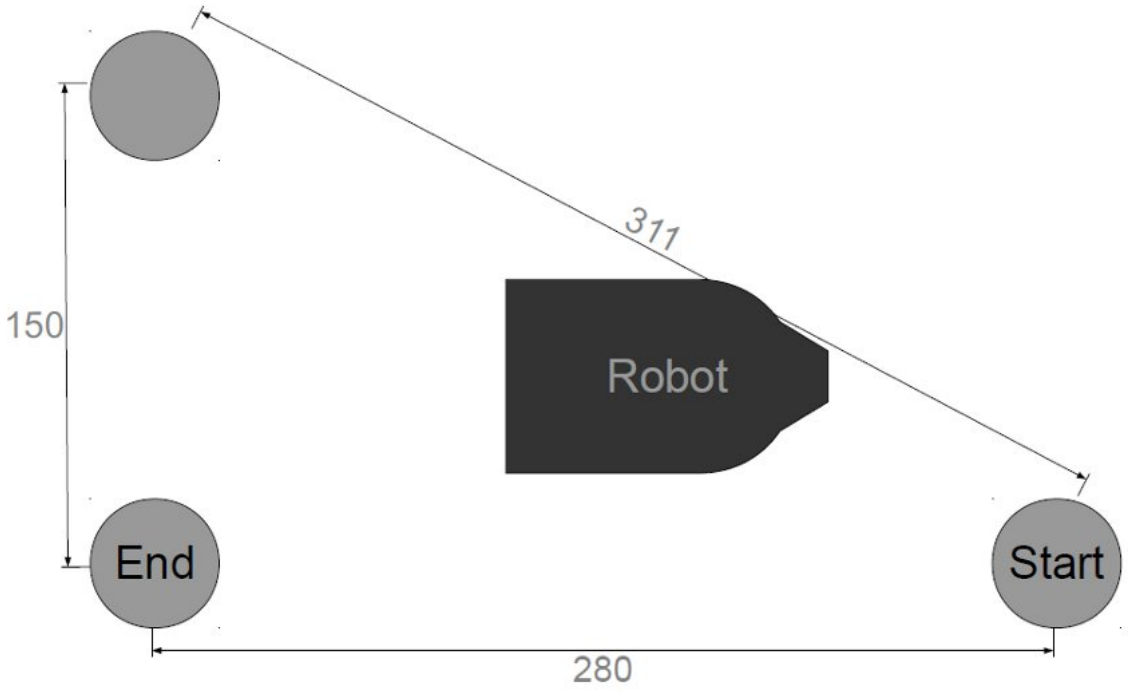


Figure A.5: The floor-plan for the towers of Hanoi setting with the mobile manipulator. Dimensions are in centimeters. The places had different heights off the ground: top left 68*cm*, low left 62*cm*, and right 76*cm*. The stacking places were 40*cm* by 40*cm* in area.

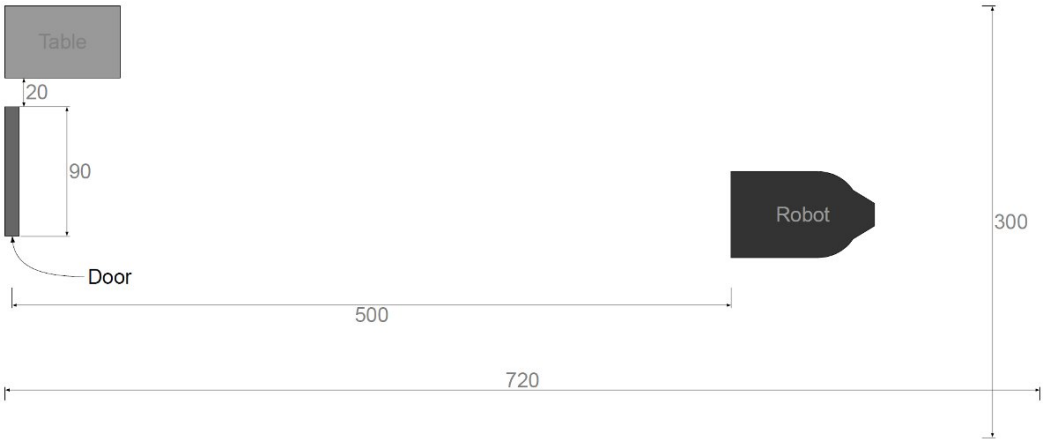


Figure A.6: Opening a door with a mobile manipulator. Dimensions are in centimeters. The Door was a common standard hallow wooden door with no closing mechanism and a turning handle.

A.5.2 Opening a Door and Exiting a Room

A major difference between all the previous cases and this particular activity is that the robot will interact with fixed bodies and surfaces. The interactions between the robot and the environment should be handled even more carefully than before as these interactions might provoke instability in the controls. In turn, the instability could cause malfunctions or result in damage to the robot or the environment. Some precautions were taken. The safety limits before shutdown for the robot were kept conservative and any force or torque was limited to a certain threshold. Moreover, the robot would shut down the arm and/or the segway if these limits were reached and exceeded.

The setting used had the robot $5m$ away from a door, facing away from it. A more detailed view can be seen in figure A.6. The door is a hallow wooden door without retracting mechanism and a turning handle lock. The door is $90cm$ wide and $2.1m$ high. The handle is positioned at a height of $1m$.

The purpose of having the robot placed away and facing the other way around from the door is to guarantee that the system allows the user to make a search of the item using only visual information, making a displacement with the mobile base and then doing coarse and fine manipulation to perform a task.