Pointing gestures for Cooperative Human-Robot Manipulation Tasks in Unstructured Environments

by

Camilo Alfonso Perez Quintero

A thesis submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science
University of Alberta

Abstract

In recent years, robots have started to migrate from industrial to unstructured human environments, some examples include home robotics, search and rescue robotics, assistive robotics and service robotics. However, this migration has been at a slow pace and with only a few successes. One key reason is that current robots do not have the capacity to interact well with humans in dynamic environments. Finding natural communication mechanisms that allow humans to effortlessly interact and collaborate with robots is a fundamental research direction to integrate robots into our daily living. In this thesis, we research pointing gestures for cooperative human-robot manipulation tasks in unstructured environments. By interacting with a human, the robot can solve tasks that are too complex for current artificial intelligence agents and autonomous control systems. Inspired by human-human manipulation interaction, in particular how humans use pointing and gestures to simplify communication during collaborative manipulation tasks; we developed three novel non-verbal pointing based interfaces for human-robot collaboration. 1) Spatial pointing interface: In this interface, both human and robot are collocated and the communication format is done through gestures. We studied human pointing gesturing in the context of human manipulation and using computer vision, we quantified accuracy and precision of human pointing in household scenarios. Furthermore, we designed a robot and vision system that is able to see, interpret and act using a gesture-based language. 2) Assistive vision-based interface: We designed an intuitive 2D image-based interface for upper body disabled persons to manipulate daily household objects through an assistive robotic arm (both human and robot are collocated sharing the same environment). The proposed interface reduces operation complexity by providing different levels of autonomy to the end user. 3) Vision-Force Interface for Path Specification in Tele-Manipulation: This is a remote visual interface that allows a user to specify in an on-line fashion a path constraint to a remote robot. By using the proposed interface the operator is able to guide and control a 7-DOF remote robot arm through the desired path using only 2-DOF. We validate each of the proposed interfaces through user studies.

The proposed interfaces explore the important direction of letting robots and humans work together and the importance of using a good communication channel/interface during the interaction. Our research involved the integration of several knowledge areas. In particular, we studied and developed algorithms for vision control, object detection, object grasping, object manipulation and human-robot interaction.

Preface

This thesis is an original work by Camilo Alfonso Perez Quintero, under the supervision of Professor Martin Jagersand. The research project, of which this thesis is part, received research ethics approval from the University of Alberta Research Ethics Board, Project Name "Communication mechanisms for cooperative human-robot manipulation tasks in unstructured environments". Pro00054665.

Chapter 4 of this thesis is based on the published and presented work:

- *Camilo Perez Quintero*, Romeo Tatsambon, Mona Gridseth, and Martin Jagersand. Visual pointing gestures for bi-directional human robot interaction in a pick-and-place task. In Robot and Human Interactive Communication (RO-MAN), IEEE 2015.

- *Camilo Perez Quintero*, Romeo Tatsambon Fomena, Azad Shademan, Nina Wolleb, Travis Dick, and Martin Jagersand. Sepo: Selecting by pointing as an intuitive human-robot command interface. In Robotics and Automation (ICRA), IEEE 2013.

- *Camilo Perez Quintero* and Martin Jagersand. Robot making pizza. In 3rd place in the IEEE Robotics and Automation Society (RAS) SAC Video contest, may, 2013.

Chapter 5 of this thesis is based on the published and presented work:

- *Camilo Perez Quintero*, Oscar Ramirez, and Martin Jagersand. Vibi: Assistive vision-based interface for robot manipulation. In 2015 IEEE International Conference on Robotics and Automation (ICRA).

- *Camilo Perez Quintero*, Oscar Ramirez, Mona Gridseth, and Martin Jagersand. Small object manipulation in 3d perception robotic systems using visual servoing. In Robot Manipulation: What has been achieved and what remains to be done? Workshop, IROS, 2014.

Chapter 6 of this thesis is based on the published and presented work:

- *Camilo Perez Quintero*, Masood Dehghan, Oscar Ramirez, and Martin Jagersand. Flexible virtual fixture interface for path specification in tele-manipulation. In Robotics and Automation (ICRA), 2017 IEEE.
- *Camilo Perez Quintero*, Masood Dehghan, Oscar Ramirez, Marcelo H. Ang, and Martin Jagersand. Vision-force interface for path specification in tele-manipulation. In Human-Robot Interfaces for Enhanced Physical Interactions Workshop, ICRA, 2016.
- *Camilo Perez Quintero*, Romeo Tatsambon Fomena, Azad Shademan, Oscar Ramirez, and Martin Jagersand. Interactive teleoperation interface for semi-autonomous control of robot arms. In Computer and Robot Vision (CRV), IEEE 2014.

Acknowledgements

My Ph.D.'s journey has been one of a kind. I'm very grateful to the University of Alberta, Edmonton, and Canada. They are fantastic places to live, do research and raise a family.

First of all, I would like to thank my advisor Professor Martin Jägersand, who has been the best example of the researcher, advisor, teacher and colleague. Thanks Martin for the unconditional support and excellent guidance.

I want to thank my supervising committee members Professors Hong Zhang, Mahdi Tavakoli, Patrick Pilarski, Simon Leonard and Elizabeth Croft for their valuable feedback on my Ph.D. proposal and research. I also want to extend my gratitude to Geoffrey Fink and Ron Kube for taking the time of reading in detail my candidacy, thesis drafts and most important for having offered a genuine friendship.

I want to acknowledge the Computing Science department for the Doctoral recruitment scholarship and various teaching assistantships, to Alberta Innovates Technology Futures for their scholarship, to the Government of Alberta for the Queen Elizabeth II scholarship and to my advisor for assistantship support. Thank you for made this research possible.

To my dear friends, thesis contributors and adventure partners Oscar, Masood, Romeo, Azad and Mona. Doing research with you has been an invaluable experience, thanks for sharing your friendship, knowledge and always have a positive face towards my research ideas. I also would like to thank my friends and colleagues in the robotics and vision lab for allowing me to learn about their exciting research and sometimes for permitting me to participate in it. Many thanks to Sepehr, Menna, Xuebin, Rong, Dana, Vincent, Abhineet, Xi, Ankush, Nina, Alejandro, Diego, Kiana, Shida, Huan, Tristan, Chris, Peter.

Finally, but certainly not least, to my loving wife, Carol and my dear son Sebastian. I am profoundly grateful for your love and patience. Thank you for being my constant motivation in life.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The current understanding of what a robot is has been around for centuries [9, 10, 11]. However, only from the late 1970's were robots massively introduced to manufacturing. Structured environments and repetitive tasks made robots succeed. Robot enthusiasts predicted a new era of robots helping humans in daily activities. Unfortunately, human environments are dynamic, difficult to structure [12] and human interaction is required.

Robotics researchers and robot companies have been struggling for more than four decades to bring robots closer to humans with only a few successful cases, e.g., robotic vacuums and robot lawn mowers. Furthermore, the idea of having a general-purpose robot that can assist and collaborate in human-like activities is still far from becoming a reality. Although current robotic platforms seem very capable, a recent unfortunate event has exposed a key problem during a robotic emergency response –*The lack of intuitive communication mechanisms with robotic platforms*. During Japan's Fukushima Daiichi power plant disaster, a highly unstructured and unpredictable environment, weeks passed before power plant personnel completed training to operate the few available rescue robots [13]. Typically, robots are instructed either by text-based programming or direct control of motions. Learning to teleoperate a multiple degrees of freedom (DOF) robot (e.g., robot arms equipped with a robot hand) is cumbersome and time consuming. As has happened in many other technological revolutions, e.g., computers, the internet, smartphones, cars, etc. *A good user interface design is essential for massive deployment and acceptance of a new technology*. Human environments require more natural communication mechanisms that allow humans to effortlessly interact with a robot.

1.2 Overview

In this thesis, we focus on communication mechanisms for human-robot manipulation interaction, specifically pointing gestures as the common interface for human-robot interaction(HRI). Our gold standard consists of building a robotic system capable of behaving as a cooperative human. Robot manipulation is a key factor to bring robots working alongside humans. Developing capable hardware and good algorithms for scene understanding, object recognition, pose recognition, grasp planning, compliant manipulation, motion planning, task planning, task execution and error detection, are only part of the collaborative manipulation challenge [14]. Without the development of good interfaces, robot integration into daily human activities is still a ways off.
An important consideration during the design of a human-robot manipulation interface is the context of the interaction. Figure 1.1 illustrates a daily manipulation task performed by a single human. Through his body as output and senses as input, the

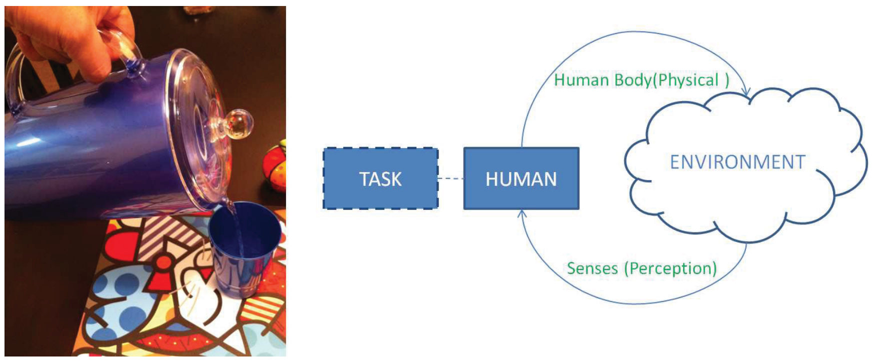


Figure 1.1: (Left) Human pouring water from a pitcher to a glass. (right) Interaction block diagram.

human pours water until the desired level is achieved. The human has learned from previous experience a sequential procedure for pouring water: (1) localize the pitcher of water and the glass. (2) Grasp the pitcher from the handle. (3) Raise the handle in a suitable relative position with respect to the glass. (4) Pour water until the glass is full by checking the water level inside the glass, and controlling the flow rate through the pitcher's inclination. To replace a human with a robot in this context, the robotic system should be able to generalize for any pitcher and glass and take into account different shapes, colors, weight of the objects and to also consider the fact that the environment is dynamic, where external entities can influence the task. To address this problem, a robotic system has to consider: dynamic and kinematic analysis for control of the robot, arm trajectory planning, object recognition, pose recognition, grasp planning, tasks execution, object avoidance, etc. The same example with a small variation is shown in Figure 1.2, where a human is pointing to a glass and a different human is later pouring water inside the selected glass. Although the

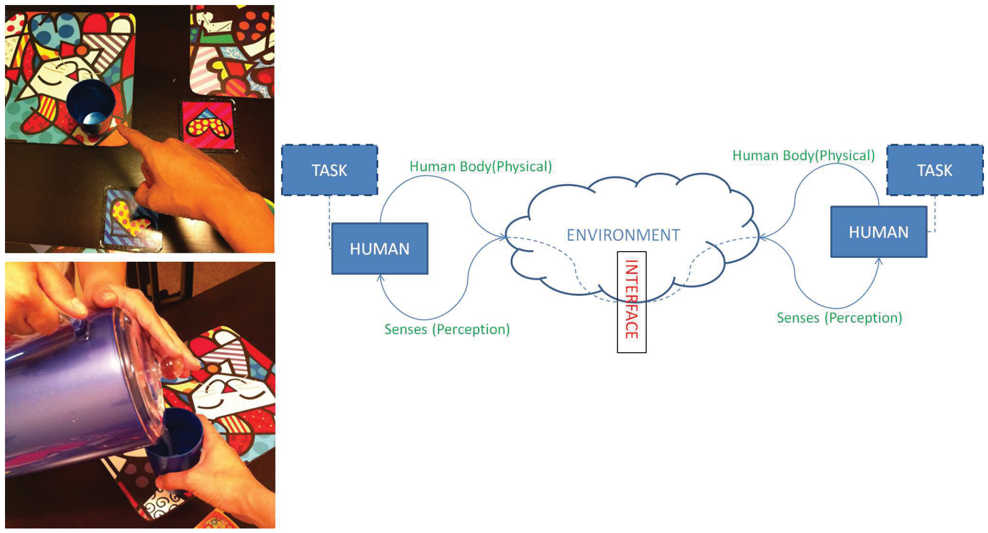


Figure 1.2: (left top) Human pointing to a glass. (left bottom) Human pouring water from a pitcher to a glass that is held by another human. (right) Interaction block diagram.

task output is the same, *i.e.,* full glass of water, including a new human in the task changes the context of the manipulation. Now, the human who is pouring the water can not only relies on his or her past experience, but also must rely on the other person's experience and their common understanding. In this case, both humans need an understandable bi-directional communication channel to collaborate and complete the task. This is represented in the diagram in Figure 1.2 by the interface block. This block represents a common interface (verbal language, gestures, symbols, etc.) that allows communication between both sides. At a first glance, replacing one of the humans by a robot seems to be a more difficult task compared to the first example (Figure 1.1). However, in our collaborative example, if instead of solving object localization and recognition for any potential glass or pitcher, we provide the robot with the capacity of interpreting human pointing. This capacity can then be used to indicate the location of any object of interest by the human. Notice that in the extreme case where the humans speak different languages, the gesture approach is still successful – The robot is now able to benefit from the human knowledge using natural interaction, through this approach many potential unsolved/complex problems can be simplified.

In this thesis we study pointing gestures as the common interface for human-robot interaction. Pointing has, arguably, universal understanding and it is the most significant type of human gesture used as a complete communicative act. Pointing can be understood as the action of directing someone's attention to something by a gesture. Inspired by human-human manipulation interaction, in particular how humans use pointing and gestures to simplify communication during collaborative manipulation tasks; we developed three pointing interactions that allowed us to study and propose novel uses of pointing gestures in the context of human robot interaction. Our research involves the integration of several knowledge areas. In particular vision control, force control, object detection, object grasping, object manipulation and human robot interaction.

1.3 Thesis Outline

The remainder of this thesis is divided into two parts, Figure 1.3 presents the organization of this thesis. Part I provides a brief introduction to the HRI field and explains different considerations to take into account when designing interfaces for interacting with a robot. Then, we provide a literature review on human pointing and how it has been used for human-robot interaction. Next, we provide basic knowledge of robot arm control and provide some methods for vision and force control which are essential for human-robot interactions. Part II is divided into three chapters that contain our contributions.

1.4 Thesis Contributions

Chapter 4: Spatial Pointing for Household Environments

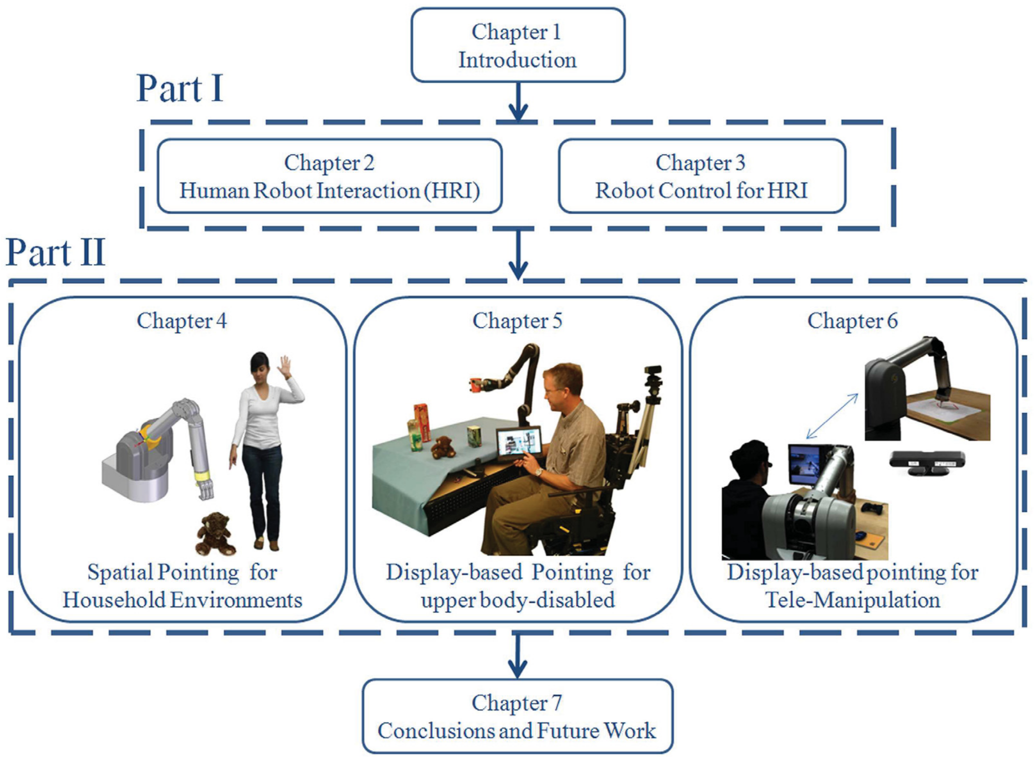

Chapter 1
Introduction
Part I
Chapter 2
Human Robot Interaction (HRI)
Chapter 3
Robot Control for HRI
Part II
Chapter 4
Spatial Pointing for Household Environments
Chapter 5
Display-based Pointing for upper body-disabled
Chapter 6
Display-based pointing for Tele-Manipulation
Chapter 7
Conclusions and Future Work

Figure 1.3: Dissertation outline.

Pointing gestures simplifies verbal communication during human interaction by providing direct spatial information of the world. Our research aims to study human pointing and use it as a human-robot interaction. Four key questions arise: Is it possible to use pointing in HRI as a core component of a simplify language to instruct and interact during a human-robot manipulation task? What complementary interactions are required to achieve non-verbal collaborative manipulation tasks? What type of human-robot manipulation task can be completed trough pointing interaction? Our research aims to quantify, evaluate, designed and implement pointing interactions for human-robot collaboration.

In this chapter, we study human pointing gestures in household environments. Using computer vision we develop an algorithm capable of interpreting pointing gestures. We use this implementation to quantify precision and accuracy of human pointing in different household scenarios and we compared it with human interpretation through user studies. We explored pick and place manipulation tasks performed by humans using non-verbal communication. Then, by using only pointing and gestures, we developed a robotic interface that permits a human to achieve pick and place tasks in collaboration with a robot, in particular we studied through user studies non-verbal bidirectional communication. To test our human-robot gesture interface we developed a study case where a user instructs a robot arm only by gesturing to prepare a pizza and then bakes it.

The main contributions of this chapter are:

- Quantitative evaluation of human pointing interpretation in a household scenario through a user study.
- Develop and testing an interface capable of interpreting human pointing.
- A robot-vision system that can: detect gestures, detect object locations on a horizontal plane, detect the human pointing direction and infer the 3D pointed location in the scene. Based on this location the system can infer which 3D object is being pointed to and return a possible 3D grasping location. If the returned object is incorrect, the system is capable of interacting with the human until the right object is found. Once found, a grasp of the desired object is performed.
- Experimental performance and an error evaluation study where eight human subjects use the robotic-system either in an instructor or assistant role in a collaborative task.
- A practical application of our system is presented where a robot prepares a customized pizza by interacting with a human by gesturing.

Chapter 5: Display-based pointing for upper body-disabled persons

Upper body disabled persons have been benefited with the introduction of wheelchair mounted arms by providing them with a way to overcome their manipulation limitations. However, current control of robot arms is performed by joystick teleoperation, which imposes challenges during mapping from the interface input DOFs to the task DOFs, producing higher cognitive load and increase the time to complete tasks. Our hypothesis is that by designing a vision-based interface that provides different levels of autonomy and reduces interface input DOFs, both cognitive load and task accomplish time will improve for the end user. Furthermore designing an interface with feedback *coming from rehabilitation manufacturers, and pilot users, provide a good insights to build a first functional system.*

We explore a display-based pointing approach by augmenting a robotic manipulation system used by upper body disabled users. Assistive robot arms can help upper body disabled persons in completing basic manipulation tasks during their daily activities. Unfortunately, high costs and poor interfaces have slowed the adoption of this technology. In this chapter, we present a vision-guided robotic system to assist upper body disabled persons with pick/place and open/close tasks. We based our design on interactions with upper body disabled users and rehabilitation experts. We validated our proposed interface through user studies.
The main contributions of this chapter are:

- Intuitive 2D image-based interface with different levels of autonomy that hides the complexity of robot arms. Our HRI has evolved through several iterations of feedback with Kinova[1] rehabilitation engineers and from interactions with our first pilot user.
- 2D grasping cursor, which allows the user to select a suitable grasp orientation.
- Two compliant modules: pulling/pushing and rotating for safe interaction of the assistive robot with the environment.
- Easy portability from one robot-arm to another. Our system can be easily integrated with any robot arm that exposes basic functionality: angular joint control and current joint values. Our system was initially implemented on a 7-DOF Barrett WAM arm, then ported to a temporarily borrowed 6-DOF Kinova Jaco wheelchair arm.
- Quantitative and qualitative evaluation of our system through a user study with our second pilot user and eight control subjects.
- Intuitions and guidelines for the development of similar interfaces.

Chapter 6: Display-based pointing in Tele-Manipulation
Events like the Fukushima Daichi power plant disaster have shown that remote robot teleoperation is still a challenge, especially when high DOF robots are required to be

[1]Kinova robotics is a Canadian company that has designed a light and capable robot arm for assisting upper body disabled persons with daily chores.

controlled. Bilateral systems are potential alternatives that provide an intuitive interface to robot teleoperation. However, it could be expensive and non-practical to provide such interfaces. Inexpensive interfaces for teleoperating a robot arm are unilateral systems, e.g., gamepad and mobile devices, but as demonstrate in Chapter 4 cognitive load and task accomplishment time increases with this type of interfaces. Furthermore, tasks requiring contact with the remote environment becomes a bigger challenge. Our hypothesis is that by providing a computer vision-based interface, the poor outcome of unilateral interfaces can be compensated to provide a similar behavior than a bilateral system. Similar to Chapter 5, our propose interface relies on provide semi-autonomy and reducing input DOFs We present a Tele-Manipulation interface that permits a user, by pointing through a display-based interface, to specify in an on-line fashion a path constraint to a remote robot arm. We validate our interface by testing it with unidirectional and bidirectional set-ups.

The main contributions of this chapter are:

- A 2D image interface that simplifies the complex process of specifying a 3D path constraint to a remote manipulator.
- An impedance control architecture that constrains the robot manipulator to follow a 3D path, while maintaining contact with the environment. The controller is specifically applicable for tasks that require contacts with the environment.
- Integration of an image interface and virtual fixtures control architecture that provides a flexible system permitting both bilateral and unilateral teleoperation with or without path constraints.
- Quantitative and qualitative evaluation of our system through a 9-participant user study, that compares the unilateral and bilateral configurations with and without the path constraints.

Part I

Background and Literature Review

Chapter 2

Human Robot Interaction

Human-Robot interaction (HRI) is a multidisciplinary sub-field of robotics that relates to understanding, designing and evaluating robotic systems for use by or with humans [15]. A misconception is that HRI only focuses on the social aspect of the interaction (user-centered) [16], how people perceive different types and behaviours of robots, how they perceive social cues or different robot embodiments, etc. These are certainly core components of the HRI field and to isolate and get a better understanding of the human factor, researchers have opted for using techniques like Wizard-of-Oz approaches –puppeteering of a robot that allows participants to envision what future interaction could be like–, to avoid technical challenges. However, without addressing the whole range of challenges from technical, cognitive/AI to psychological, social cognitive and behavioural, the end goal of integrating robots into human environments is far from being completed. In this chapter and Chapter 3 we provide background for addressing both user-centred and robot-centred approaches in the context of pointing gestures for cooperative human-robot manipulation tasks. At the end of this thesis, we expect to provide as part of our contributions, real world functional robotic systems with user-centred design interfaces.

This chapter is organized as follows. Section 2.1 presents important considerations for designing human-robot interactions. Next, Section 2.2 presents a review of human-robot pointing gestures.

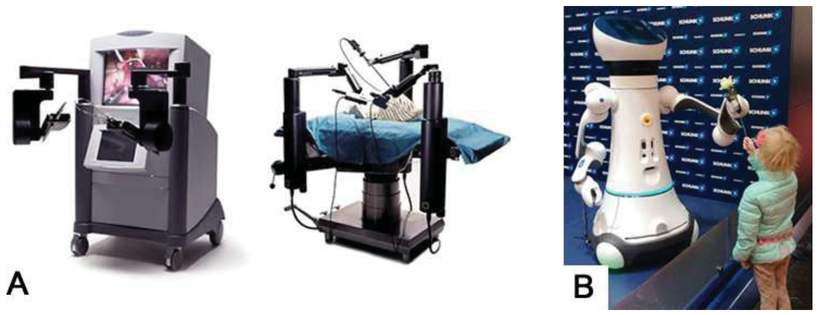


Figure 2.1: **A**. Zeus system used in trasatlantic surgery [1]. **B.** Fraunhofer Care-O-bot 4 service robot interacting with a human.

2.1 Design Factors for HRI

As robots start appearing more in daily life, human-robot interaction (HRI) becomes a need. Arguably, there is not a clear procedure for how to design a good HRI. Perhaps the most common approach used by researchers consists of studying a similar human-human interaction. Next, extract a simplified model that can be replicated with a robotic system, and then, evaluate via human subject studies [17], [18]. In this section we present factors that are important to consider during the design of human-robot interactions [15].

2.1.1 Human Proximity

Depending on the human and robot location, the interaction can be classified as proximate or remote interaction. In a proximate interaction human and robot are collocated, *e.g.,* a service robot interacting with a user in the same room. In the remote interaction human and robot are separated spatially or even temporally. A remote interaction with a physical manipulator is often referred to as telemanipulation; some examples are a transatlantic surgery [1] (see Figure 2.1) and the Canadarm2 teleoperation [19]. Notice that during the remote interaction all the information received by the user comes from the system interface. In this case the designer has to guarantee that the transmitted information is enough for the user to accomplish the task. In contrast, in the proximate interaction, the designer can take advantage of the extra information that the system is not exposing, but that the user is perceiving from the environment.

2.1.2 Types of Robot

A common way to categorize robots is by the type of locomotion and kinematics. Common classifications for locomotion and kinematics are: cartesian, cylindrical, spherical, articulate (robot arms), wheeled, walking, slithering, underwater, etc. In the last two decades robot platforms have taken a dramatical change in their mechanical and sensory capabilities, by including in their design, the fact that in their working environment humans are now taking a central role. In contrast with the rigid position-controlled robots used in industry, current trend in human-friendly robot manipulators are designed to have very lightweight structures and very low power consumption, *e.g.,* the Barrett WAM arm [20], DLR lightweight arm [21], Kinova Jaco arm [22], etc. Due to this new design paradigm where interaction with humans is a requirement, a more recent robot classification used in the context of physical human-robot interaction (pHRI) [23] is based on the robot's capability of interaction: *supportive, collaborative and cooperative. Supportive robots* aim to provide useful information and may assist the human in achieving a particular task, *e.g.,* museum tour guided robots, shopping assistant robots, airport assistant robots, etc. This robots normally presents a display based interface that outputs the required information. Designers need to pay special attention to prevent collisions and generate suitable proxemic behaviours. The *collaborative robots* work together with the human to achieve a particular task. This task is divided into subtasks and human and robot works separately on different subtasks that best suite their abilities. Normally the human is in-charge of sub-tasks that require decision making or high dexterity and the robot performs repetitive or precision type tasks. The third type of interaction is the *cooperative*. This involves direct physical contact between the human and the robot or indirect contact through a common object, *e.g.,* rehabilitation therapy, kinesthetic teaching, etc. In this thesis will be working mostly with *collaborative and cooperative* interactions.

2.1.3 Level and behaviour of autonomy

During the design process, the level of autonomy is dictated by the context of interaction and both, robot and human capabilities. One of the first literature reports discussing levels of autonomy is presented by Sheridan *et al.* [2] in the context of underwater robots. Figure 2.2 shows a graph between levels of autonomy and task entropy (capability). They proposed different levels of autonomy where the lower

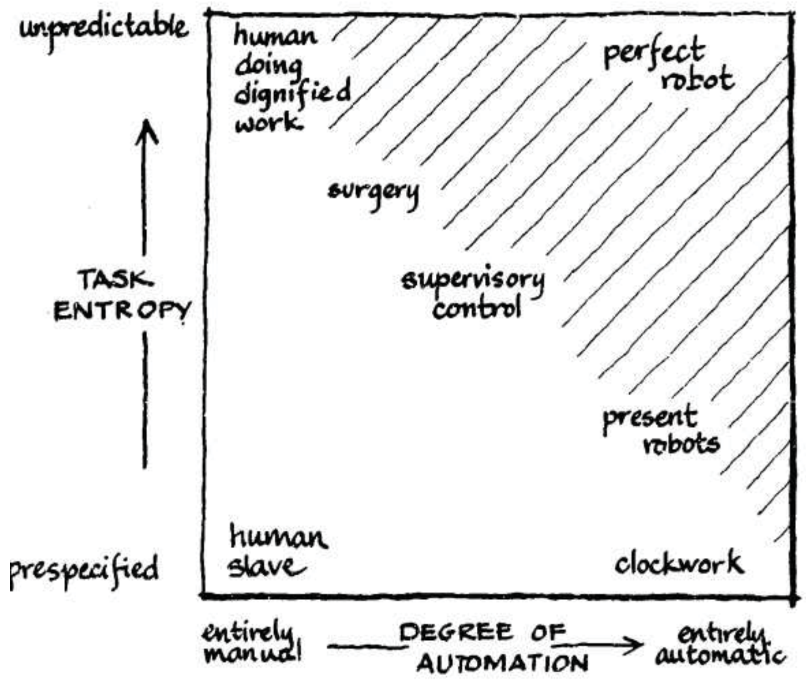


Figure 2.2: Graph taken from Sheridan and Verplank [2]. Automation vs. Entropy.

level is complete teleoperation of the robot and the highest level consists of the robot performing a complete task autonomously. From a different perspective even an autonomous system may have some sort of high level control from the human in the sense of being programmed. Goodrich *et al.* [15] presents a scale of levels of autonomy with emphasis on human interaction (see Figure 2.3). In contrast to Sheridian, in Goodrich's scale, the highest level is peer-to-peer collaboration instead of autonomy. The challenge here is to create robots with the appropriate cognitive skills to interact naturally and efficiently with a human. In order to keep the interaction natural, the robot must be able to behave with full autonomy at certain times and may need to support social interactions. In practice a balance needs to be reached to find a level of autonomy suitable for the task with realistic capabilities of the automation and the need to actively engage the human user [24]. Notice that the scale also defines when the robot is an *extension of the human* user/operator (teleoperation, mediated teleoperation) and when it becomes a *separate entity* (collaborative control, peer-to-peer collaboration).

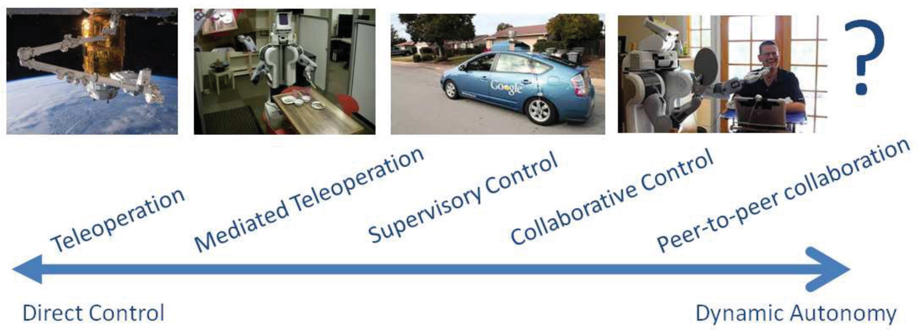


Figure 2.3: Levels of autonomy for human-robot interaction. From left to right: Canadarm2, Heaphy project: aimed to recruit people online and train them to operate PR2 robots to carry out everyday tasks, Google autonomous car, Robots for Humanity project [3].

2.1.4 Information Exchange

There are two variables that specify the information exchange during a human-robot interaction: (1) the communication medium and (2) the format of the communication. The former refers to how the information is presented. Typical mediums are: visual displays, gestures, speech, non-speech audio, physical interaction and haptics (See Figure 2.4). The latter refers to how the information is structured through the medium, *e.g.,* subset of language (speech), ecological displays (visual display), traditional windows-type interaction (visual display), warning through vibrations (haptic), etc. Another interesting approach is the multi-modal interfaces in which several inputs are merge in the same interaction. For instance, a human can give instructions to another human on how to assemble a piece of furniture. During these instructions she or he may use both speech and gestures to increase understanding. From the implementation point of view, it may be easier to get all the information through one channel. However, from the human point of view, a human-like communication is the ideal. Common metrics used to quantify efficiency of the interaction are the interaction time, cognitive or mental workload, shared understanding or common ground between human and robot and situation awareness produced by the interaction.

2.1.5 Context of the Interaction

The design of the interface is highly influenced by the application area and the context of the interaction. For example, the designer of a space tele-manipulation interface has to take into account microgravity effects and the delay between the robot system

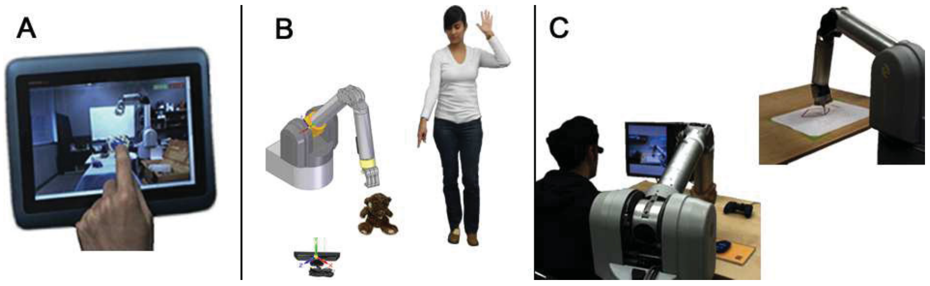


Figure 2.4: **A.** Screen coordinate interface. The user specifies through a touch screen what object to grasp. **B.** Gesture based interface. The user commands the robot to pick up an object by spatial pointing. **C.** Haptic teleoperation. The user in the master module can feel the feedback from touching the table top while writing through the slave robot.

at the remote site and the human operator. Studying and understanding the context of the application is a requirement to provide a good interface. Some examples of application areas where robotics have penetrated during the last decade are: rehabilitation and health care robotics [25], space robotics [26], disaster robotics [27], domestic robotics [28], agriculture and forestry robotics [29], underwater robotics [30], etc.
As part of our contributions, in chapter 5, we developed an interface for upper body-disabled. In the next section, we present an introduction to the rehabilitation robotics field with emphasis on assistive robotics for upper body-disabled persons. For more details in other of the aforementioned application areas, please refer to the corresponding cited references.

2.1.6 Rehabilitation Robotics

The field of rehabilitation robotics is commonly subdivided into therapeutic robotics and assistive robotics. The former focuses on patients in the process of recovering physical, cognitive or cognitive functionalities. The latter, assist users with disabilities to achieve daily activities. Therapeutic applications normally require the person in recovery and a therapist who is familiar with the therapy robot. The main advantages of using a robot in this context are long periods of therapy without getting tired, better tools to measure recovery, which encourages the user to continue the therapy and engaging therapy exercises (e.g, computer game-based).

Therapy robots can be classified by the contact point with the patient. The end-effector-based therapy consists of attaching the human hand or foot to the robot's end effector, the therapy movement is performed in a Cartesian space. In contrast, the exoskeleton-based therapy robots also restrict the joint movements. This permits sensing the specific configuration of the human and assisting each human joint independently. In these robots the designer has to pay special attention in terms of weight, dimensions, and wearability. The third type of robots are the non-contact therapy robots, which focuses on monitoring and coaching the patient. In contrast with rehab robotics, that are intended to be used temporarily, assistance robots need to be designed to be usable for long periods. The designer needs to involve and take into account users preferences in order to gain acceptance. Assistance robots can be classified by manipulation, mobility or cognition. Manipulation aids can be presented as a fixed platform (e.g, assisting in the kitchen, desktop bed) or a portable platform, *e.g.,* a robot arm mounted on an electric wheelchair or an independent assistive robot capable of navigating and manipulating objects in the environment. Mobility aids are electric wheelchairs with navigation systems and motorized walkers. The cognitive aid assists people with dementia, autism or other mental limitation. In the next section we focus on assistive manipulation aids performed with robot arms mounted on electric wheelchairs for a comprehensive review in other assistive technologies please consult [25].

2.1.6.1 Mounted Wheelchair Robot Arms

Assistive robotics can help people with movement disabilities. In particular, robot manipulators can benefit people with limited upper body mobility to perform every day tasks, see Figure 2.5. Unfortunately, the adoption of these kind of technologies have been limited, mainly due to the complexity of robot manipulation tasks. State of the art rehabilitation robot arms try to duplicate human arm functionality by providing 6 or 7-DOF motion. Typically manufacturers provide a joint or Cartesian based control using a joystick interface or a customized input device based on the end user's abilities. Users typically suffer from disabilities such as spasms, where the disabled cannot stably hold objects, and or muscular dystrophy making control a bigger challenge. Mapping from the high DOF arm to a 2-DOF joystick input device results in switching between modes, that couple the joystick to different translation, rotation and grasp motions (see Figure 2.6). This is time consuming, cumbersome and increases complexity and cognitive load for users. While direct joystick control is the norm in rehabilitation, a lot of robotics research focuses on autonomy. User interaction can be minimized by ceding more autonomy to the robot.

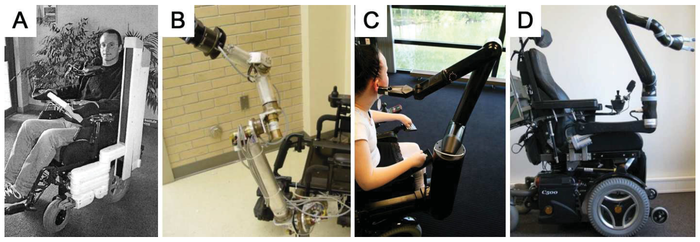


Figure 2.5: Wheelchair mounted arm examples: A) Weston wheelchair, based on SCARA. B) WMRA-I University of South Florida. C)iArm, Exact dynamics. D) Jaco arm, Kinova robotics

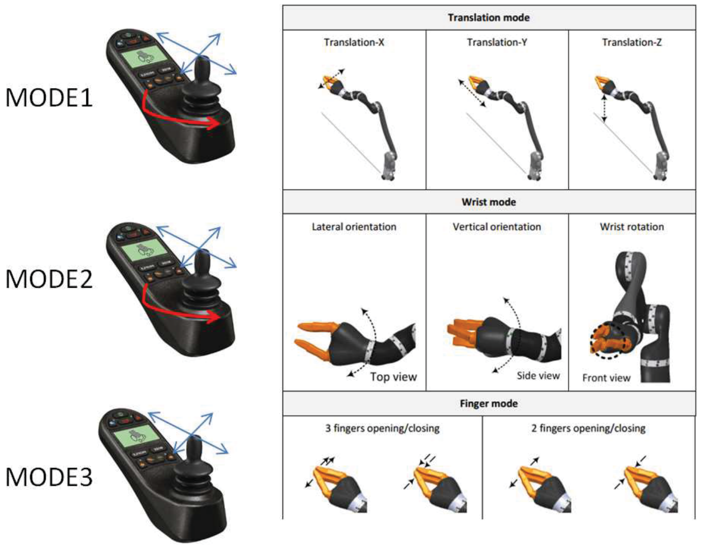


Figure 2.6: Jaco Joystick modes. Image courtesy of Kinova robotics.

In the past decade several researchers have been working on improving the interface and interaction between upper body disabled users and assistive robot arms. A common approach has been through vision systems. Notable examples are reviewed here. Driessen *et al.* [31] present a collaborative controller, in which the Manus ARM [32] is equipped with an eye-in-hand-camera, a distance sensor and a display. The system allows the user to select a coloured object and reach towards it. A notable result as they report at the end of their experiments is that an eye-in-hand-camera does not give a natural point of view for the human to interact with.

Tsui *et al.* added a shoulder camera that provides a perspective of the wheelchair occupant for the interface. Based on [33], Tsui divided the process of picking an object into two steps: first a gross reaching motion to a location close to the object, and then a fine adjustment of the hand pose. Their early work, [34] focusses on the gross motion, proposing a selection method in which the shoulder image view is divided into four quadrants which the user can iterate over. The user selects the quadrant which contains a majority of the desired object and the robot centers on it. Then, the process is repeated inside the selected sub-quadrant. Later in [35], they reported, that through an evaluation of their prototype interface, their quadrant method was found to be difficult for cognitively impaired users and that a fixed camera view outperforms a moving camera for object selection. Their next work [7], presented a system that can autonomously retrieve a desired object from a shelf. Their system set-up consisted of: Manus ARM, one touch-screen, two stereo cameras, and pressure sensors inside the gripper. After object selection, using the shoulder stereo camera, the system computes the distance to the object and commands the robot to get close to it. Next, using visual servoing, the gripper orientation is aligned by matching the eye-in-hand camera view with a previously recorded image database template.

A similar set-up (Manus ARM, eye-to-hand camera, stereo eye-in-hand camera) was used by Dune *et al.* [36], [37]. However, their approach does not need a priori 3D models of the objects nor an image database. After clicking over the object in the eye-to-hand camera view, the eye-in-hand camera uses visual servoing for scanning the epipolar-line and detects the location of the object by image processing. After detection, they used stereo virtual visual servoing or active estimation of the object shape for the autonomous object grasping.

So far we have focused our review on works using wheelchair mounted arms. However there are also numerous mobile manipulators used as assistive robot research: SAM, Care-O-bot 3 [38], EL-E [39], SAM [40], PR2 [41], etc. A drawback with these systems, as pointed out by [41], is that despite previous advances in assistive mobile manipulators, none has been widely adopted by disabled end users to date. Part of the reason is their cost as many of these mobile manipulators robots are not suitable for widespread commercial adoption. Another limitation is the use of an allocentric system where the robot can move independently from the user. For instance, Ciocarlie *et al.* [41] test their system with a pilot user completing a task that comprises of retrieving a towel from the kitchen including navigation, door/drawer opening and closing, object grasping. The complete task took 54 minutes. In contrast, the wheelchair mounted arm option, by having the human-in-the-loop in a physical egocentric reference, reduces operation time and solves many of the perception, navigation and manipulation problems encountered with assistive mobile manipulators.

2.2 Pointing for HRI

The ultimate aim is to bring Human-Robot Interaction (HRI) to a point where interactions with robots will be as natural as interactions between humans. To this end, incorporating gestures in HRI is an important research area. Gestures are intended to convey meaningful information. Linguistic researchers normally classified human gestures in five categories [42]: **Deictic** gestures are generally understood as pointing gestures and refers to spatial locations, *e.g.,* objects, places, etc. **Iconic** gestures depict some feature of the action or event described. **Metaphoric** gestures are representations of abstract ideas but the form of the gesture comes from a common metaphor, *e.g.,* showing an empty palm hand may indicate a problem. **Beat or motor** gestures are normally used to keep the rhythm of a conversation, *e.g.,* marking initiation of new discourse or introduction of new sub-topics. **Symbolic or emblematic** gestures are normally cultural dependent some examples are waving, “thumbs up”, shaking the head as a “yes” or “no” gesture.

2.2.1 Gesture Recognition

Gesture recognition is a broad field that that covers many research areas. Typically the recognition pipeline can be structure into three steps: detection, tracking and recognition. Figure 2.7 shows a classification for the three steps involved in the recognition process. The detection step is in charge of isolating the relevant data (human upper limbs, hands, head, face, body skeleton, etc.) from the image background. Some common features used to isolate the relevant data are skin color, shape, motion models, anatomical models etc. After detection, the next step is tracking the region of interest. Some examples of tracking methods are: (1) Template based tracking, which invokes the detection algorithm near the spatial vicinity that the region of interest was detected in previous frame. (2) Optimal estimation consist in turning feature detection into estimations, a common framework is the Kalman filter. (3) Particle filter, where the location of the region of interest is modelled with a set of particles and the location of the region of interest is given in a probabilistic way. The last step is recognition. If the gesture is static a general classifier can be used. However, when the gesture is dynamic, a temporal aspect has to be taken into account. Common techniques are k-means, k-nearest neighbour, support vector machine, hidden Markov model, finite state machine, etc. For more technical detail please refer to [43], [4].

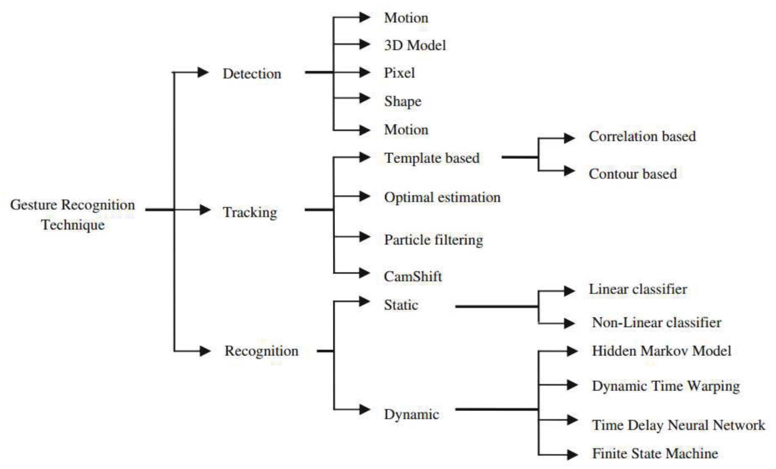


Figure 2.7: Recognition techniques classification, image taken from [4]

2.2.2 Pointing Gestures in Robotics

Arguably the most significant type of human gesture used as a complete communicative act is pointing. Pointing is present in all known human societies [44] and some communication researchers have concluded that pointing is an important stage in linking pre-verbal communication and spoken language [45]. Pointing can be understood as the action of directing someone's attention to something by a gesture [46]. When a human instructs another to perform a task, non-verbal communication cues in the form of pointing and gesturing play important roles. For instance Lozano *et al.* [47] conducted experiments where human participants performed an assembly task instructed by speech or pointing gestures. Participants understood and learned better from gesture-only than from speech-only instructions. As shown by Lozano *et al.*, non-verbal gesture communication is a powerful human-human interaction. Pointing gestures provide only coarse spatial information of a target location. However, in a human-human pointing interaction, the recipient must be able to determine what the intention of the communicator in directing to some location is. Researchers have studied pointing gestures in different robotic contexts.

We present a classification based on participant(s) and the direction in the interaction: **Human pointing**, **human pointing to robot**, **robot pointing to human**, **human and robot interacting by pointing**.Note that for the cases where a robot is part of the interaction, accuracy and reliability requirements are more relevant as opposed to a pure human pointing analysis. Table 2.1 shows a list of related work classified based on the human-robot pointing interaction it also points out which studies integrates manipulation as part of the interaction.

2.2.2.1 Human

The virtual reality (VR) community has extensively studied pointing interaction for selecting objects in a virtual environment. Some common pointing-based interactions are:

- **Simple ray-casting technique** [61]: A virtual representation of a user's hand is created in a virtual environment by tracking his hand position and orientation. From the tip of the virtual hand, a ray defines the direction of pointing; more than one object can be intersected by the virtual ray, but only the closest object to the user is selected.

- **Two-handed pointing** [62]: Both user's hands are tracked and used to defined the virtual ray.

- **Flash-light technique** and aperture technique [63]: this technique imitates pointing at objects with a flashlight, instead of using a virtual ray, it uses a conic selection volume. A drawback is disambiguation of the desired object when more than one object falls into the spotlight. The aperture technique allows the user to interactively control the width of the selection cone.

VR pointing techniques have the advantage of interacting with a virtual world, where target sizes can be designed more arbitrarily, and failure has no physical consequence. When dealing with real world objects, more precise methods are required. Human machine interaction researchers have focused on building interfaces capable of detecting pointing gestures and estimating the pointing direction. Kahn and Swain [48] introduced a pointing gesture detection method. They designed the Perseus architecture based on image segmentation where the pointing direction is found only in the 2D image. Jojic *et al.* [49] developed a system for detecting 3D pointing gestures and found an estimation of the gesture direction using stereo cameras and dense disparity maps in real-time. Nickel and Stiefelhagen [50] designed a system that detects pointing gesture and estimates the 3D pointing direction in real time. They performed face and hand tracking and then used hidden Markov models to classify the 3D trajectories in order to detect the occurrence of the gesture. The system integrates color and depth information in a probabilistic framework to obtain

Table 2.1: Pointing classification

Interaction	References	Context or Application	Robot Manipulation
Human	[48]	Pointing and detecting trash on the floor.	No
	[49]	Controlling a cursor on the screen by pointing/selecting targets in a room.	No
	[50]	Selecting targets on the floor.	No
	[51]	User can light regions of a virtual reality scene by pointing.	No
	[52]	The robot has predefined locations in a known building. The user points to a particular direction and the robot informs what is there.	No
Human $\rightarrow$ Robot	[53]	The user points to a soda can and the robot picks up the can.	Yes
	[54]	Pointing to fixed targets.	No
	[55]	By pointing the user gives spatial directions to a robot.	No
	[56]	Pick up selected household objects	Yes
Human $\leftarrow$ Robot	[57]	The robot was placed in a public environment pointing to a fixed location. Participants were asked to interpret what the robot was pointing at.	No
	[58]	Robot narrator using gestures to enrich its speech.	No
	[59]	Handling dangerous liquid with a robot.	Yes
Human $\leftrightarrow$ Robot	[60]	The human selects a color plate by pointing to it. The robot points to the same plate and verbally says the plates color.	No

3D-trajectories of head and hands. Kehl and Van Gool [51] present a multi-view approach that measures 3D directions of one or both arms. Their approach is based on tracking user's eyes and fingertips and then extracting body silhouettes from multiple views and uses point correspondences to reconstruct the 3D points of interest. Park *et al.* [64] present a real-time 3D pointing gesture recognition algorithm. They used a stereo camera and particle filters for tracking hands and face. They also used two hidden Markov models to discriminate pointing gesture from other gestures to estimate the pointing direction. A different approach based on a monocular camera is presented in [52], where previous knowledge of pointing locations has been assumed and pointing directions have been estimated by fusing hand pointing gestures and user's head pose.

2.2.2.2 Human Pointing Gestures to a Robot

Waldherr *et al.* [53] proposed a vision-based gesture interface for human-robot interaction. They used an adaptive color-filter, a pose template matcher and a neural network based method for recognizing human arm gestures. They tested their system by integrating it into a mobile robot that performs a clean-up task. Some limitations of their proposed system are that the person's face needs to be visible all the time and the user is required to stay within a distance range from the robot. Using a Time-of-Flight camera, Droeschel *et al.* [54] applied hidden Markov models to detect pointing gesture and proposed a learning approach were a model of human pointing is learnt. The learning process consists of a Gaussian process regression that inputs body features and output the pointing direction. They tested their system on a mobile manipulator platform by fetching an object located on top of a table. Van den Bergh *et al.* [55], used pointing gestures to direct a mobile robot where to go. They used a Kinect sensor for hand detection and, by tracing a vector between the wrist and hand location, they find the pointing direction. They used a classifier to discriminate between open, fist or pointing gesture. The point-and-click method proposed by Kemp *et al.* [56] enables humans to select a 3D world location using a laser pointer. The robot is then able to detect the laser spot and estimate its position with respect to the robot.

2.2.2.3 Robot Pointing to Human

Williams *et al.* [57] investigated how humans interpret robot pointing. They used a stationary PR2 robot pointing to a particular location/object. Human participants were asked to identify what the robot was pointing at. Their results showed that people used the robot arm direction to interpret robot pointing behaviour. However, when robot's head faced the same direction as the pointing arm, the majority of the participants used the head direction to infer the pointing. Chien *et al.* [58] studied gestures to improve human-robot interaction in a narration scenario. Their results showed that robot deictic gestures were particularly effective in improving information recall in participants. In [59], Ende *et al.* studied relevant gestures for co-working tasks. They implemented the gestures in a robotic system and evaluated how well their proposed gestures are recognized by humans. They presented a use-case where a human hands over an empty container to a three arm robot. The robot task is to fill the bottle with a potentially dangerous liquid and close it. During the task execution, two robots are working on the task and a third is in charge of interacting with the human. If the human gets close to the robot during the task execution, the robot performs a warning gesture. When the task is finished the robot performs a calling gesture and points to the closed container.

2.2.2.4 Human and Robot Interacting by Pointing

Rich *et al.* [60] study human-robot engagement by developing a pointing interaction where a robot asks a human to point to a plate. After the human's pointing gesture, the robot confirms which plate the human was pointing to. Since the focus of their research is on engagement and collaboration, they simplified the robot vision problem as much as possible, making it useful only for particular cases. They used an adaptive view-based appearance model tracker for face, gaze and head nod detection and a color tracker to implement plate and hand tracking.

Chapter 3

Robot Control for Human Robot Interaction

During human-robot interaction, control techniques play a fundamental role. In this section we present an overview of robot control techniques suitable for HRI.

3.1 From Human to Robot Vision Control

The majority of actions that we humans perform on a daily basis involve vision. For manipulation, we constantly use our vision to locate, manipulate and reorient objects. A misconception is to think of the eyes as passive receivers, instead, during a manipulation task, the eyes are active and tightly coupled, temporally and spatially, to the motor actions of the task. Even for a highly repetitive/automated activity (*e.g.,* preparing tea), Land *et al.*[5] experimentally showed that during tea preparation, the eyes closely monitor every step of the process (see Figure 3.1). They concluded that even highly automated activities involve both checking and closed-loop control. Furthermore, when an action has become "automatic", it is not just the motor acts themselves that have become automated. The complete control systems is responsible for their execution, which include sensory elements such as proprioceptors, touch receptors, and eyes.

Hayhoe *et al.*[6] study the visual closed-loop control during a common manipulation task (sandwich preparation, see Figure 3.1). Their research focuses on finding what scene information do observers actually need, and how much of this information persists past a given fixation. They concluded that much of natural vision can be accomplished with "just-in-time" representations. However, during their experi-

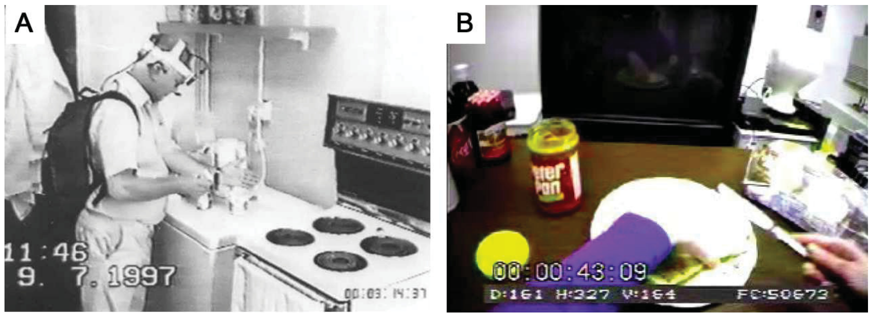


Figure 3.1: **A.** Setup for the tea experiment, taken from [5]. **B.** Setup for the sandwich experiment. The participants were asked to make a peanut butter and jelly sandwich and to pour a glass of soda. Image taken from [6].

ments they observe several aspects of performance that point to the need for some representation of the spatial structure of the scene that is built up over different fixations. Patterns of eye-hand coordination and fixation sequences suggest the need for planning and coordinating movements over a period of a few seconds. This planning must be in a coordinate frame that is independent of eye position, and thus requires a representation of the spatial structure in a scene that is built up over different fixations.

If we were going to build a simplified version of the human visual control system behaviour, two main requirements are needed: (1) A visual closed-loop control that can work with "just-in-time" representations and (2) from time to time a scene spatial structure. The former can be achieved with a well known control technique, visual servoing. the latter with point cloud perception. Both approaches are explained in the next two sections.

3.1.1 Visual Servoing

In the context of robot manipulation, visual servoing uses visual information to control the pose of the robot's end-effector relative to a target object or a set of target features [65].

This target object or set of target features are used to define a control error function, which has to be regulated to zero from a initial state (*e.g.,* current end-effector location) to a desired state (*e.g.,* end-effector desire pose), see Figure 3.2. Depending on how the target object or target features are defined, visual servoing can be classified to three main approaches: position-based visual servoing [66], image-based visual servoing [66] and 2D 1/2 visual servoing [67].

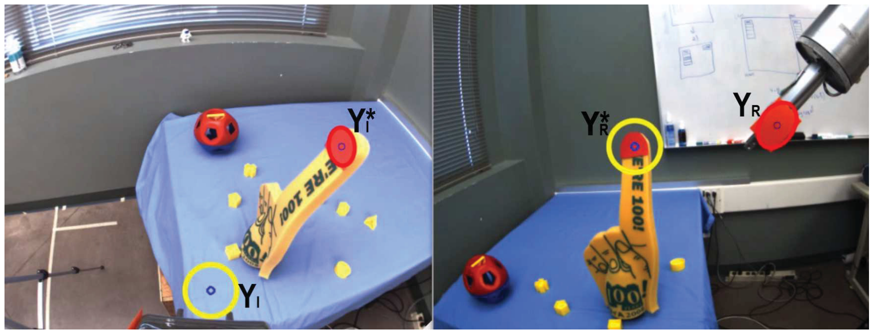


Figure 3.2: Left: eye-in-hand camera. Right: eye-to-hand camera. In this example the error function is defined in the image space by $\binom{Y_l}{Y_R} - \binom{Y_l^*}{Y_R^*}$

- Position-based visual servoing (PBVS): Image features and object geometrical information are used to estimate the relative pose between the camera and the object. With this information the desired pose is calculated and the error is defined between the current and desired pose of the robot in the task space, Figure 3.3.

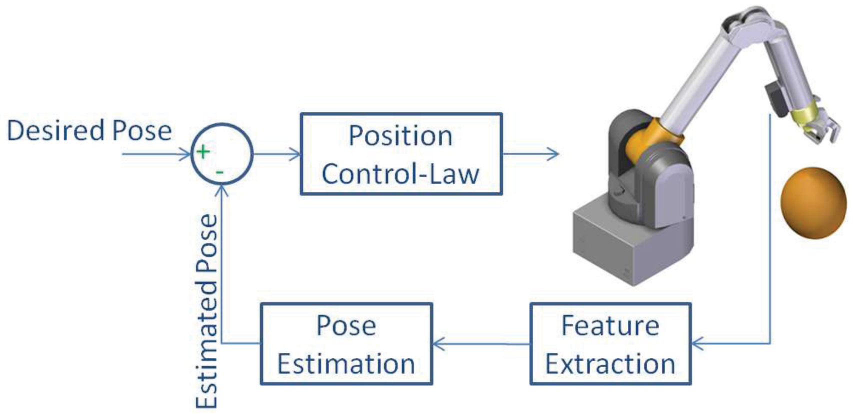


Figure 3.3: Position-Based Visual Servoing (PBVS) block diagram. PBVS has the advantage of described a manipulation task directly in cartesian pose. However, feedback depends on system calibration parameters which turn PBVS highly sensitive to calibration error.

- Image-based visual servoing (IBVS): In contrast with PBVS where the error is defined in the task space coordinates, IBVS defines the error directly in the image space, Figure 3.4. In contrast with PBVS, IBVS is considered to be very robust with respect to camera and robot calibration errors. Coarse calibration only affects the rate of convergence of the control law in the sense that a longer time is needed to reach the desired position [65].

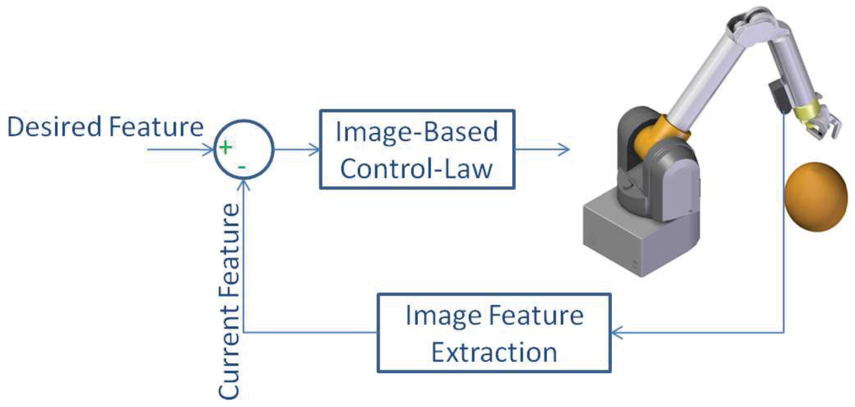


Figure 3.4: Image-Based Visual Servoing (IBVS) block diagram. The difference between desired and current image feature is calculate and input into the IBVS controller.

- 2D 1/2 visual servoing: Both image and 3D data are used to specified the error, Figure 3.5. In contrast with PBVS this method does not required the object model. The method is based on the estimation of the camera rotation and scale translation between the current and the desired views of the object by estimating a Euclidean homography (mapping between points in two Euclidean planes) transformation between views. The homography can be written in terms of internal camera parameters, camera displacement between the views and the equation of the plane being viewed [68].

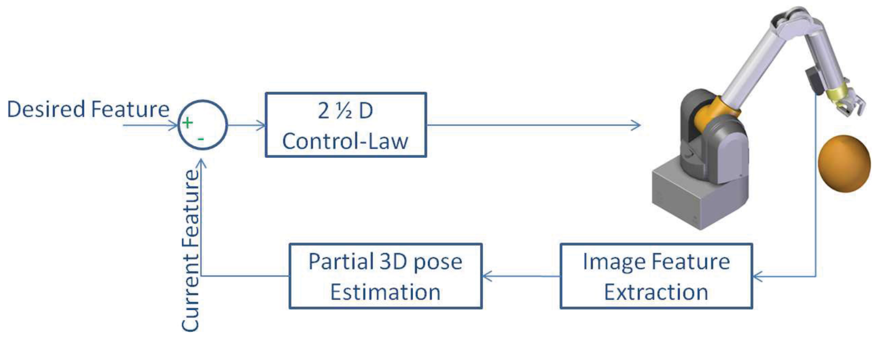


Figure 3.5: 2D 1/2 Visual Servoing block diagram. Both 2D and 3D data is used. However in contrast with PBVS the object model is not required, instead the 3D data is estimated through two camera views.

Depending on the visual servoing approach (IBVS,PBVS, 2D 1/2 VS) and the task, the vision data may be acquired from a camera mounted on the robot's end-effector (eye-in-hand-camera) or from a fixed camera in the workspace so that the robot motion can be observed (eye-to-hand). The number of cameras may varies also depending on the servoing approach and the task. For more technical detail please refer to [65, 69, 70, 71].

Unfortunately as pointed out by Tatsambon *et al.* [72, 73], substantial work has been published in visual servoing over the years, but only few real world applications have been accomplished. A notable example evaluated with upper body disabled persons is presented by Tsui *et al.* [7]. In their system to operate the robot arm, the user selects an object from the display. Two inputs are present: touch screen or mouse emulation. After clicking on the desired object the system calculates the position of the object using stereo vision from a shoulder camera. The robot arm moves to a close position. With the object in view, the robot arm computes a more precise target by using the Scale Invariant Feature Tracking (SIFT) descriptor [74] to estimate the 3D position for the target object from the gripper stereo camera. The procedure consist in using epipolar geometry to match SIFT features from both cameras and then based on those points fit a plane and find a normal vector. This normal vector is used to refine the pose of the robot end effector. The next step consists on recognizing the object by matching SIFT features with the template image database. It is possible that the object is not recognized if the current view is a partial view, e.g., the object is viewed from the side whose template was taken from the front. Finally a 2D 1/2 visual servoing scheme is used to align the current gripper camera view with the pertinent template database image, See Figure 3.6.

So far we have reviewed classic approaches to visual servoing and described an application example of wheelchair mounted arm system that relies on it to achieved object fetching from a shelf. An interesting approach for unstructured environments is the uncalibrated visual servoing (UVS) approach. As opposed to the above described methods, UVS studies vision-based motion control of robots without using the camera intrinsic parameters, the calibration of the robot to camera transformation, or the geometric object/scene models. This is a demanding problem with increasing applications in unstructured environments, where no prior information is assumed [75, 76, 77].

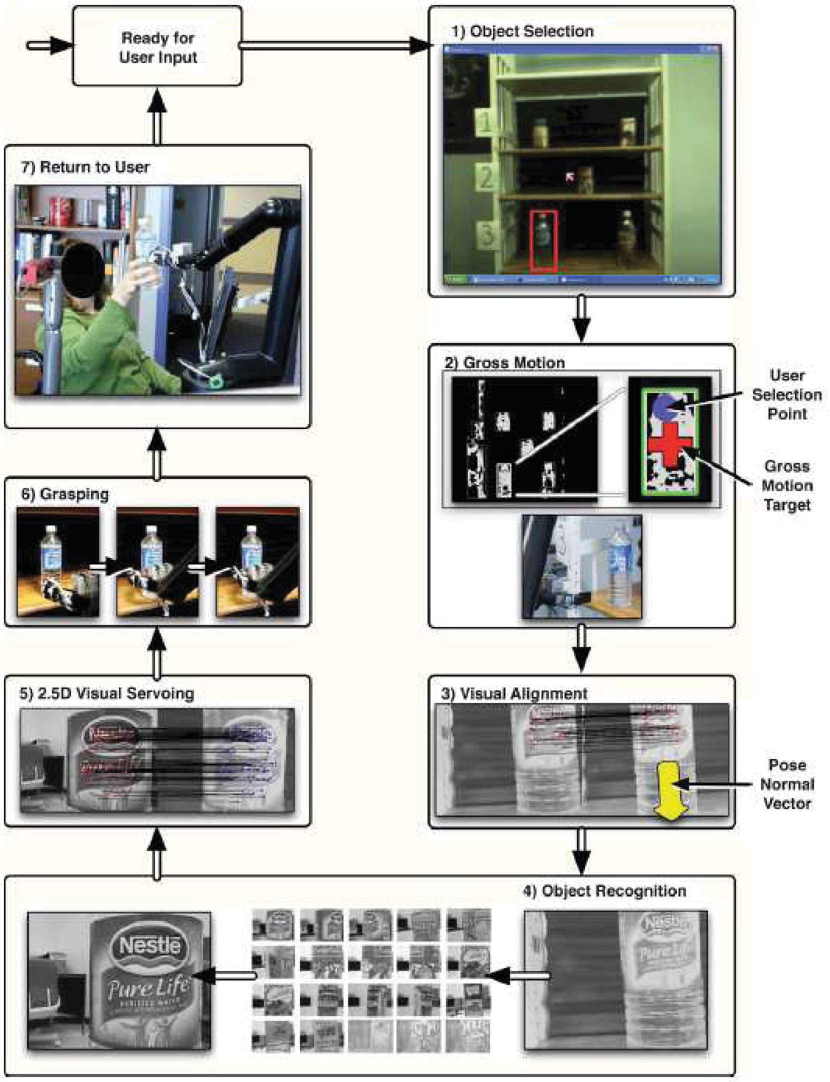


Figure 3.6: Operational diagram from Tsui *et al.* paper [7]

3.1.1.1 Uncalibrated visual servoing

The control law in UVS should be defined without the need to reconstruct depth or other 3D parameters. One way to define the uncalibrated control law is an approach similar to IBVS. Let $F : \mathbb{R}^N \to \mathbb{R}^M$ be the mapping from the configuration $\mathbf{q} \in \mathbb{R}^N$ of a robot with N joints, to the visual feature vector $\mathbf{s} \in \mathbb{R}^M$ with M visual features. For example, for a 6 degrees of freedom (DOF) robot with 4 point features (8 coordinates in total), $N = 6$ and $M = 8$. The visual-motor function of such vision-based robotic system can be written as

$$\mathbf{s} = \mathbf{F}(\mathbf{q}). \tag{3.1}$$

This formulation is general and covers both the eye-in-hand and eye-to-hand systems. The time derivative of the visual-motor function in (3.1) leads to

$$\dot{\mathbf{s}} = \mathbf{J_u}(\mathbf{q})\dot{\mathbf{q}}, \tag{3.2}$$

where $\dot{\mathbf{q}}$ is the control input and $\mathbf{J_u} = \frac{\partial \mathbf{F}(\mathbf{q})}{\partial \mathbf{q}} \in \mathbb{R}^{M \times N}$ is called the *visual-motor Jacobian*. The discrete-time approximation of (3.2), when $\mathbf{J_u}(\mathbf{q})$ is replaced by $\widehat{\mathbf{J}}_\mathbf{u}(\mathbf{q})$ is:

$$\Delta\mathbf{s} \simeq \widehat{\mathbf{J}}_\mathbf{u}(\mathbf{q})\dot{\mathbf{q}}. \tag{3.3}$$

Similar to the IBVS control law, the estimated visual-motor Jacobian, $\widehat{\mathbf{J}}_\mathbf{u}$, appears in the uncalibrated control law:

$$\dot{q} = -\lambda \widehat{\mathbf{J}}_\mathbf{u}^{+}(\mathbf{s} - \mathbf{s}^*), \tag{3.4}$$

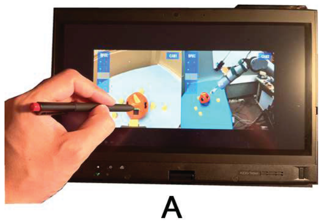
where λ is a rate constant, $\widehat{\mathbf{J}}_\mathbf{u}^{+}$ is the Moore-Penrose pseudoinverse of $\widehat{\mathbf{J}}_\mathbf{u}$ and s^* is the vector containing the desired values of the features. In the control law (3.4), the visual-motor Jacobian $\widehat{\mathbf{J}}_\mathbf{u}$ is estimated from data. Different methods of estimation exist, for example the orthogonal exploratory motions method [78], the Broyden method [76, 79], the least-squares based method [75]. A natural choice is the Broyden method for its simplicity. This method can be summarized as follows

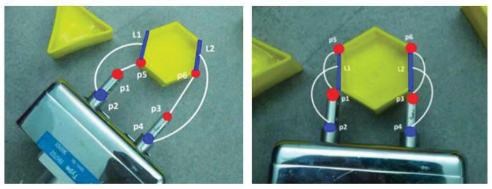
$$\widehat{\mathbf{J}}_\mathbf{u}^{(k+1)} = \widehat{\mathbf{J}}_\mathbf{u}^{(k)} + \alpha \frac{\left(\Delta\mathbf{s} - \widehat{\mathbf{J}}_\mathbf{u}^{(k)}\Delta q\right)\Delta q^\top}{\Delta q^\top \Delta q}, \tag{3.5}$$

where α is the forgetting factor which is used to lessen the weight of old data during the estimation process. The initial guess $\widehat{\mathbf{J}}_\mathbf{u}^{(0)}$ of the visual-motor Jacobian can been estimated using orthogonal exploratory motions method.

3.1.1.2 Task Specification

In IBVS an easy way to interact with the robot during a manipulation task is through visual task specification, see Figure 3.7A. A visual task can be defined with the objective of bringing the end-effector of the robot arm to its target in the work space [80]. For example point-to-point and point-to-line tasks align a point on the robot end-effector with a point and line respectively in the workspace. These are


A



B

Figure 3.7: **A.** Visual task specification through a touch interface **B.** Specification of point-to-point and point-to-line task to align a gripper with the hexagon and close the gap to grasp it.

simple tasks that can be specified in an interface by having the user click on end-effector and target points on the screen showing the robot camera view [73]. Much of the work that a robot arm could do for the user requires more complex tasks. We can think of these complex tasks as being composed of several simple tasks. The interface should make it easy for the user to compose complex tasks out of simple ones. In the example in Figure 3.7B we illustrate how one could combine point-to-point and point-to-line tasks in two steps to align an end-effector with a hexagon and then close the gap.

The tasks that the user specifies in the image consist of 2D image information. Even though the end-effector converges to its goal in image space, that does not guarantee that the robot achieves the task in 3D world space. This means we need to be able to verify that the tasks specified by the user will in fact perform as expected. Depending on the level of calibration there exist tasks that can be unambiguously specified and verified using two 2D image views. They are said to be decidable tasks [80, 81]. For example point-to-point tasks are decidable for uncalibrated systems. It was shown by Dodds *et al.* [81] that there exists operators that can compose simple decidable tasks into more complex, but still decidable tasks. Therefore we can use simple decidable tasks to construct useful high-level tasks for robot. We can create an interface that lets the user combine decidable tasks in an intuitive way to instruct the robot visual servoing.

Looking back at Figure 3.7B, we can get a set of equations that will allow us to use these tasks for visual servoing. The two point-to-line tasks between p2 and L1 and p4 and L2 will align the gripper with the hexagon and maintain the correct orientation throughout the process. The point-to-point tasks p1 to p5 and p3 to p6 will first bring the gripper close to the hexagon as seen in the developement from the left to the right image. Finally they will allow the gripper to close the gap to the hexagon so the grasp can be completed. We assume a stereo camera system, where L and R subscripts represent the left and right camera views respectively. Using image coordinates provided through the user interface the point-to-point and point-to-line tasks are represented as follows:

$$E_{pp}(p, p') = (p'_L - p_L, p'_R - p_R)^T \tag{3.6}$$

$$E_{pl}(p, L) = (p_L \cdot L_L, p_R \cdot L_R)^T \tag{3.7}$$

where p, p' and L represent the general point and line coordinates. We can combine the different tasks and preserve decidability by using the AND function defined in [81]. This allows us to stack the equations for several tasks that should hold simultaneously to give us the following error vector for the scenario in figure 3.7B:

$$E = (E_{pp}(p1, p5), E_{pp}(p3, p6), E_{pl}(p2, L1), E_{pl}(p4, L2))^T \tag{3.8}$$

The vector E becomes the error function that we need to minimize in the visual servoing control law. Hence to complete the task we start with the situation in the left image and minimize E to servo the gripper to its location in the right image. Finally we update the locations of p5 and p6 and complete the task.

3.1.2 Point Cloud Perception

Recently emerging low cost depth sensors give researchers the opportunity of working directly with 3D data. The simplest data in three-dimensional Euclidean space is a 3D point p_i. A collection of 3D points is normally referred to as point cloud $\mathcal{P}$. The most common 3D sensing approaches are: (1) Time-of-Flight (TOF): The physical distance between the sensor and the target object or surface is estimated by measuring the time it takes for a emitted signal to hit a surface and return to the receiver. (2) Triangulation techniques: The 2D cameras extrinsic and intrinsic parameters are known and used to recover depth information. (3) Structured light sensors: A pattern is projected, normally in infrared light, and a sensor uses the pattern information to calculate depth. A few years ago (November 2010) the Microsoft Kinect was released using this approach. Figure 3.8 shows a point cloud collected using a Microsoft Kinect.

Having the 3D point information is not enough to achieved a manipulation task. Normally a meaningful data extraction is required. As seen in Figure 3.8 besides

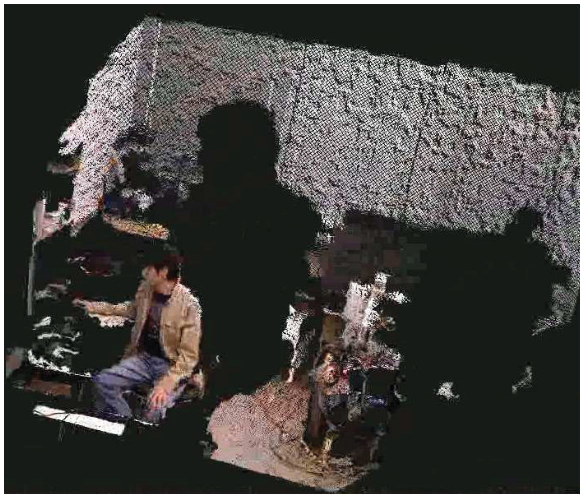


Figure 3.8: Both RGB and depth sensor are used to generate a point cloud visualization with color.

the 3D euclidean coordinate $p_i = x_i, y_i, z_i$ the cloud data can hold more information $p_i = f_1, f_2, f_3...., f_n$ where f_i represents a feature value, *e.g.*,color, label, geometry, etc. Normally spatial decomposition techniques, such as kd-trees or octrees are used to speed queries of a particular neighbouring region $\mathcal{P}^k$. In robot manipulation, clustering and segmentation is needed to identify the objects to be manipulated. Below is an introduction to some basic approaches to get meaningful data from a point cloud.

3.1.2.1 Plane extraction

In human environments objects are likely found on top of planes, using this contextual knowledge helps simplify the grasping space of the robot. Rusu [82] presents a simple algorithm for identifying planes in a point cloud $\mathcal{P}$:

1. Randomly select three non-collinear unique points pi, pj, pk from $\mathcal{P}$.
2. Compute the model coefficients from the three points $(ax + by + cz + d = 0)$;
3. Compute the distances from all $p \in \mathcal{P}$ to the plane model (a, b, c, d);
4. Count the number of points $p^* \in \mathcal{P}$ whose distance d to the plane model falls between $0 \leq |d| \leq |d_t|$ where d_t represents a user specified threshold.

The above is repeated k times and the set points p^* are stored. The set with largest inliers is selected as the plane. The planar model coefficients are estimated in a least-squares formulation, and a bounding 2D polygon (*e.g.,*2D convex hull) can be estimated as the best approximation given the input data of the supporting horizontal plane.

3.1.2.2 Object clustering on top of planes

After identifying the plane and its boundaries, the next step is to cluster the points on top of the plane in different possible objects. This can be achieved by:

1. Creating a kd-tree representation for the input point cloud dataset $\mathcal{P}$;
2. Setting up an empty list of clusters C, and a queue of the points that need to be checked Q;
3. For every point $p_i \in \mathcal{P}$, perform the following steps:
   - Add p_i to the current queue Q;
   - for every point $p_i \in Q$ do:
     - search for the set $\mathcal{P}_i^k$ of point neighbours of p_i in a sphere with radius $r < d_{th}$;
     - for every neighbour $p_i^k \in \mathcal{P}_i^k$, check if the point has already been processed, and if not add it to Q;
   - when the list of all points in Q has been processed, add Q to the list of clusters C, and reset Q to an empty list
4. The algorithm terminates when all points $p_i \in \mathcal{P}$ have been processed and are now part of the list of point clusters C.

3.1.2.3 Finding Edges

In a 2D image, edges are usually identify by calculating the image gradient. In 3D, edges can be identify by abrupt geometry changes. The thresholding of points having high curvature values can be performed by analyzing the distribution of curvatures in a given point cloud $\mathcal{P}$.

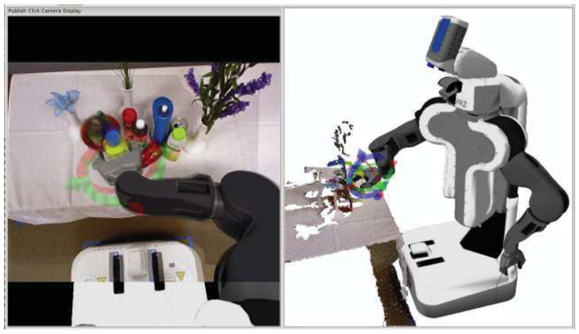


Figure 3.9: A point-and-click interface proposed by Lepper *et al.*[8].

3.1.2.4 Segmentation via Region Growing

A simple approach to segment regions consist of modifying the Euclidean cluster algorithm presented above finding and segmenting connected regions in $\mathcal{P}$ with smooth surfaces that are separated by edges at step 3. The algorithm would start by adding a point p that belongs to $\mathcal{P}$ with a low curvature value to a queue, and verify each of its p_k neighbours to see whether it could belong to the same region as p. The criteria used to identified whether it could belong to the same region is to calculate n and n_k which are the surface normals at p and p_k. Then if the angle between the normals is less than a set threshold the point p_k is added to the current region started from p.

Above we gave a quick overview of point cloud perception and in particular we presented some examples of clustering and segmentation techniques. For a complete introduction to point cloud-based perception refer to [82]. An interesting project using point cloud perception has been presented by Lepper *et al.* [8]. Their system was initially used for general remote human-in-the-loop grasp execution and later they integrated in the "Robots for Humanity project" [3]. The user interface provides the user with two displays (Figure 3.9): live images from the robot's camera and a virtual camera showing a 3D model of the robot and the point cloud visualization. The user controls the robot through 3D widgets attached to the 3D robot model in both of the user interface visualizations.

In the case where the objects can be segmented an autonomous routine can be executed [83]. The system inputs are 2D images and noisy 3D point clouds of the scene. The system performs a plane detection and a euclidean clusterization for object detection. The centroid of an object can be found from the clusters by using a principal component analysis (PCA) to compute the axes that have the min and max variance. Then the bounding boxes of the point cluster along those axes are found.

Their method uses three heuristic grasp approaches, top, side and high point grasps and ranks them using small set of features weights. **Top Grasps:** For both x and y a grasp is generated centred above the bounding box of the object. Depending if the object fits in the gripper multiple grasps are generated along z axis. **Side Grasps:** The gripper is moved in the horizontal plane along the x or y axes. If the size of the bounding box along the x-axis fits inside the gripper, the x-axis is aligned with the gripper closing direction, while y is used as approach direction. Multiple grasps are generated by sampling along the z-axis. **High point grasps:** Set of points are chosen at random within 2cm of the top of the cluster. The approach direction is z axis and the gripper is aligned with the line that connects the chosen point with the bounding box center, this approach can be used *e.g.,* to grasp bowls or other rotationally symmetric containers with rims. For objects that the robot is unable to segment or recognize, the user can used the 3D widgets from the interface to specify a final pose and allow the robot to autonomously plan a collision-free path for pickup. For more arbitrary tasks, such as pushing objects or opening doors and drawers, the user can directly control the arms by moving the 3D widgets.

Part II

Contributions

Chapter 4

Spatial Pointing for Household Environments

Pointing to indicate direction or position is one of the intuitive communication mechanisms used by humans in all life stages. It is commonly used when humans communicate spatial directions and actions while collaboratively performing manipulation tasks. In this chapter, we study and develop mechanisms for human-robot interaction using pointing gestures to achieve collaborative manipulation tasks.

4.1 Introduction

In recent years, robots have started migrating from industrial to home assistive scenarios. One of the biggest challenges in this transition is finding natural communication mechanisms that allow humans to effortlessly interact with a robot. The most significant type of human gesture used as a complete communicative act is pointing. As humans, we interpret pointing in the rich context of prior information and other cues, but rarely reflect on the actual accuracy of pointing. Encoding verbal clues into location information is non-trivial and often entails increased complexity at the software level [84]. Non-verbal pointing communication can be a low-cost alternative in the Human-Robot Interaction (HRI) context. A challenge in integrating pointing gestures is that pointing direction cannot be easily inferred by a third person or a robot.

Our hypothesis is that by providing to a robot the capacity of interpreting and performing pointing, these can be utilized as a core communication component to accomplish collaborative manipulation tasks with a human counterpart. Furthermore,

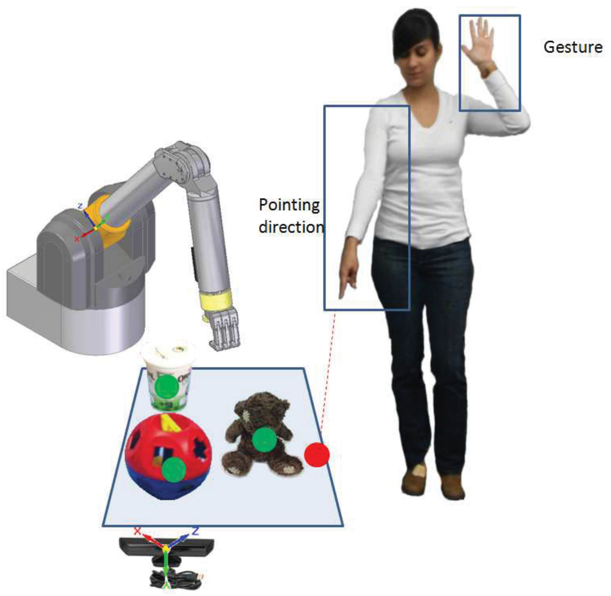


Figure 4.1: A human interacts with a robot by pointing to a particular object. The robot is able to interpret human pointing and give feed-back to the human to confirm which object she is pointing to.

we think that the capacity of interpreting pointing gestures can influence the performance during collaborative tasks. Our aim is to use human pointing interpretation accuracy as reference point to design a robot vision system capable of interact with a human through pointing and gesturing 4.1. To the best of our knowledge there is no a quantitative report of human pointing accuracy and precision for household tasks. Therefore, a first step in our research consists on quantitative estimating human capacity on interpreting pointing gestures. We perform three experiments to study precision and accuracy of human pointing in typical household scenarios: pointing to a “wall”, pointing to a “table”, and pointing to a “floor”. In average we found that human pointing interpretation has an accuracy of 16.6 cm. Based on that result, we develop an interface for selecting by pointing in a 3D real-world situation. The user points to a target object or location and the interface returns the 3D position coordinates of the target. Experimental results show that the proposed interface interprets pointing more accurately than humans in the same household scenario. The proposed interface enables users to point and select objects with an average 3D position accuracy of 9.6 cm.

As a second step we design a robot vision system and integrate our pointing interface on it. Our robot system is able to see, interpret and act using pointing and

symbolic gestures. The former can be defined as directing the attention of a recipient spatially to something in the surroundings (deictically). The latter is attempt to induce others to imagine things by gesturing iconically, that is, by pantomiming. However, many human gestures are conventional and language dependant, e.g. shaking the head as a "yes" and "no" gesture. Several studies [46] suggest that humans acquire and use these conventionalized gestures in basically the same ways they acquire and use linguistic conventions. In our research, we focus on the pointing gesture because of its, arguably, universal understanding. However we include some symbolic gestures for completeness in the interaction. We compared by experimentation three interactions: human-human, human-robot and robot-human for solving a pick-and-place task. In particular we study: human and robot pointing interpretation and human and robot visual feedback. We prove experimentally that our robot vision system can exhibit behaviour similar to and, in specific cases, better than a human in a collaborative pick and place task based on pointing and gesturing.
Finally to illustrate our gesture interface in a real application, we let a human instruct our robot through gesturing to collaborative make a pizza by selecting different ingredients.
In summary the main contributions in this chapter are 1. Quantitative findings on human pointing interpretation accuracy during human-human collaborative manipulation tasks in household environments. 2. The design of a robot vision system capable of interpreting human pointing. 3. Experimental validation and estimation of the system's accuracy and precision through user studies. 4. The design of a robot vision system capable of interact through pointing and gesturing to complete pick and place tasks. 5. Experimental evaluation of the system through user studies, in particular, we quantify and compare the importance of feedback during human-human, human-robot collaborative manipulation interactions.

4.2 Pointing gestures through a pick-and-place task

To study the pointing interaction between human and robot, we design a pick-and-place task example application. It consists on clearing objects off a table and sorting the objects in the appropriate containers. In the task we have two actors: instructor and assistant, and two types of communication: non-feedback and feedback. The actions performed by the actors for both types of communication are described in

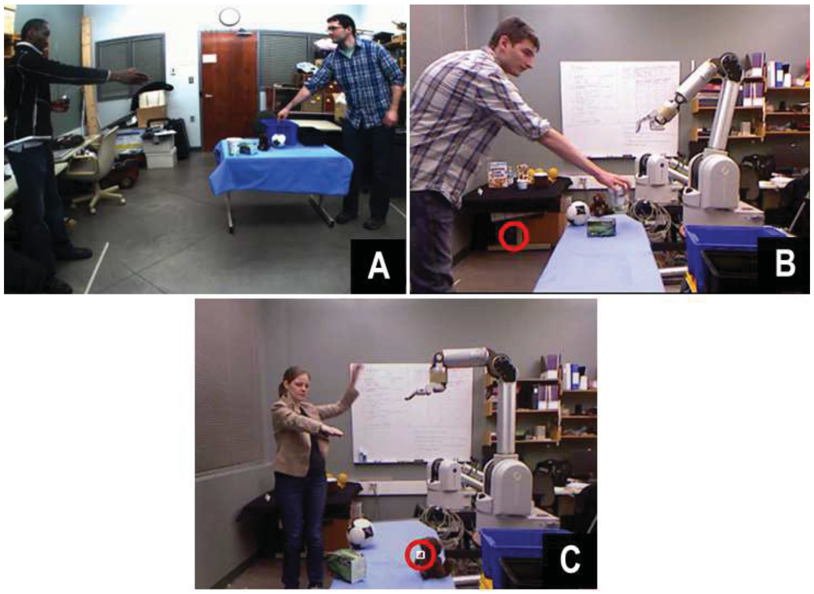


Figure 4.2: Instructor and assistant interaction
A. Human-Human: Human instructor selects an object by pointing. A human assistant asks if his interpretation is correct.
B. Robot-Human: Robot instructor selects an object by pointing. A human assistant interprets the pointing gesture performed by the robot, approaches the table and picks the selected object.
C. Human-Robot: Human instructor selects an object by pointing. A robot assistant is ready to reach for the selected object.

the work-flow diagram shown in Figure 4.3. We are covering human-human, robot-human and human-robot interaction (see Figure 4.2). Below are the steps for the pick-and-place task in the feedback communication case.

1. Instructor points to a desired object.
2. Assistant visually interprets the pointing and in return points to the object she thinks is the desired one.
3. Bi-directional interaction between instructor and assistant to either confirm the selection or refine the pointing location.
4. Instructor points to drop location.
5. Assistant drops object in the desired location (Here we skip the robot feedback for shortening the experiment time).

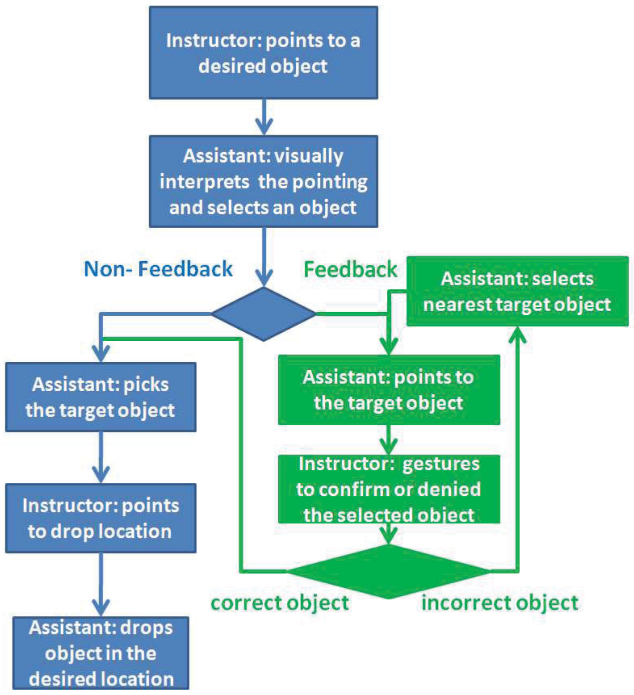

Instructor: points to a desired object
Assistant: visually interprets the pointing and selects an object
Non- Feedback
Feedback
Assistant: selects nearest target object
Assistant: picks the target object
Assistant: points to the target object
Instructor: gestures to confirm or denied the selected object
Instructor: points to drop location
correct object
incorrect object
Assistant: drops object in the desired location

Figure 4.3: Pick-and-place task work-flow diagram. Blue section: interaction without feedback. Green section: interaction with feedback.

The remainder of this chapter proceeds as follows. First, a description of the human-robot and the robot-human system is provided. Then experiments and findings are described.

4.3 System Description

Based on the pointing gestures our system provides a two-way communication channel between a human and a robot. We study this interaction through the previously described pick-and-place task application. An example of our system working in the human-robot case with feedback is in Figure 4.4 (the red ring indicates the pointing location detected from the depth image projected into the RGB image). For the non-feedback case the B,C and D steps shown in Figure 4.4 are removed. Besides the pointing gesture, we include the Yes and No symbolic gestures, both for robot-human and human-robot interaction. The particular expression of the symbolic gestures is not important for the pointing evaluation here. They could, for instance, be replaced by verbal Yes and No.

Below is a description of our human-robot and robot-human system. For both

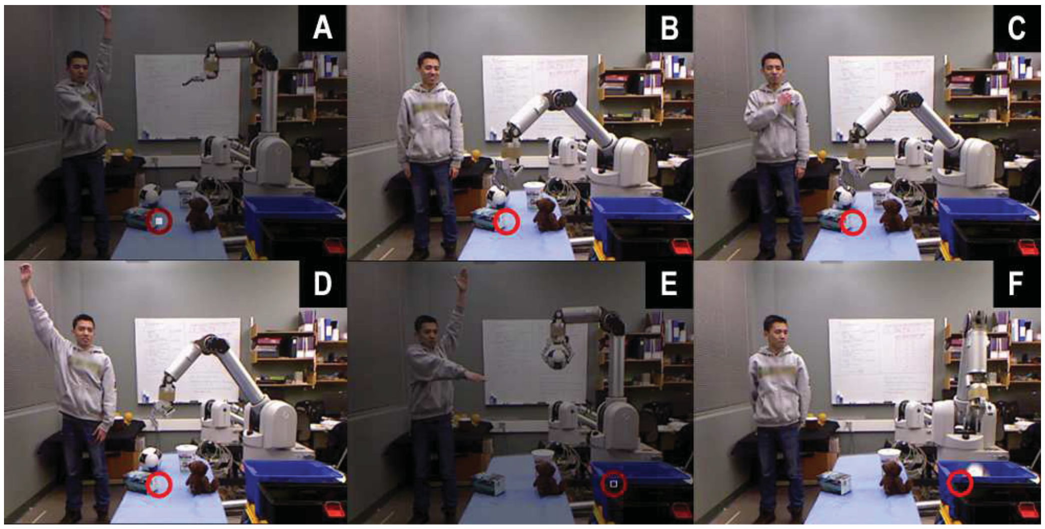


Figure 4.4: A human-robot gesture sequence interaction with our system
A. Human instructor selects a desired object by pointing —- "Pick the object that I'm pointing to".
B. Robot assistant interprets the pointing gesture performed by the human and performs a confirmation gesture to the human instructor —- "Is this the object that you want?".
Notice the pointing gesture performed by the robot arm.
C. Human instructor denies robot interpretation by crossing his dominant arm over his torso —- "No".
D. Robot assistant moves to the next possible selected object —- "Is this the object that you point?".
Human instructor confirms the robot interpretation by raising his dominant hand —- "Yes".
E. Robot assistant picks up the selected object. Human instructor selects a desired dropping location —- "Drop the object in the blue container".
F. Robot assistant places the object in the selected container.

interactions, our system uses a 7DOF WAM arm, a Microsoft kinect sensor and a regular Linux machine. Our software implementation uses ROS [85], the Point Cloud Library [86], OpenCV library [87] and OpenNI skeleton tracking libraries [88].

4.3.1 Human-Robot system

A block diagram of the human-robot system is shown in Figure 4.5. Humans use vision as input and the human body for interaction. Our robot system uses a kinect[1] as the input device to interpret human gestures and the robot arm to render gestures to the human. We assume that the human and the objects are in the field of view of our depth camera and the possible selected objects are located in a region that the robot can reach. Figure 4.7A shows a point cloud visualization of our system, not seen by the user, but shown here to illustrate the system implementation. A user points to a desired object. Notice the virtual red line that is generated using the red sphere in the users head and the teal sphere in the users hand. After the hit location is found (red sphere at the end of the virtual ray) the system corrects the hit location to the nearest object and proposes a possible grasping location on top of the object (dark blue sphere). Figure 4.7C shows the identified objects in the system (green spheres) and the grasping location (dark blue sphere). The system gets ready to give feedback to the user.
Our system is composed of four blocks: Selection, Gesture Identification, Object Localization and Decision (Figure 4.5).

4.3.1.1 Selection

The Selection or selecting by pointing (SEPO) [89] block allows the user to point to a particular location and through a gesture inform the system that the target direction is defined. Having defined the direction, the system calculates the pointing location with respect to the camera location. Figure 4.6 shows the SEPO block diagram which can be simplified in two principal stages: (a) Gesture and Pointing Identification Algorithm (blocks 1-5 in Figure 4.6); (b) 3D Point Hit Algorithm (block 6 in Figure 4.6).

We used the Microsoft Kinect sensor for acquiring depth and color image data (blocks 1 and 2 in Figure 4.6). The first step is to perform color and depth camera calibration (block 3) [90]. Then using, the free cross-platform Kinect driver OpenNI and the NITE skeletal tracking library [88], we identify the upper-torso joints (block 4). These joint locations are the input to the spatial gesture recognizer module (block 5), which is implemented as a finite-state machine. In our particular pointing

[1]the kinect sensor provides RGB and depth data

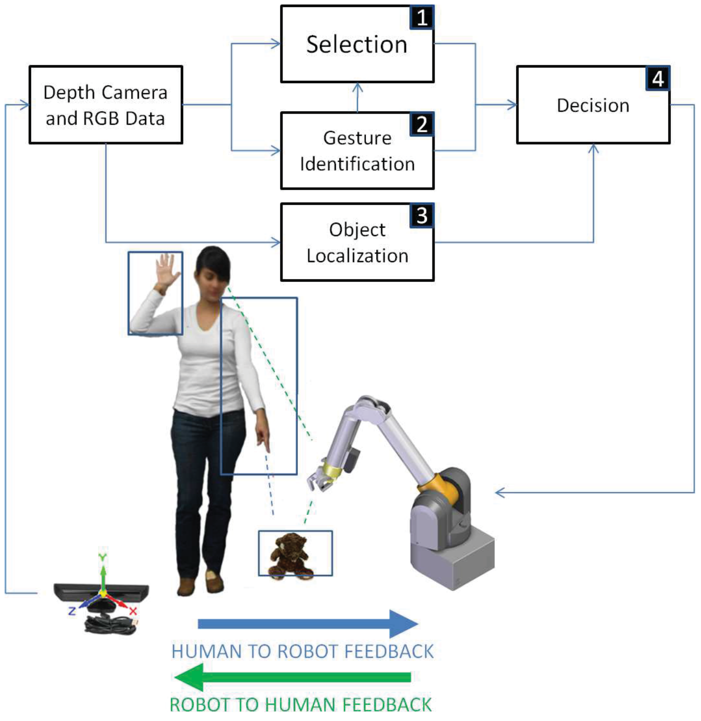


Figure 4.5: System Diagram. Through the human gesture identification, object localization and pointing interpretation:
*The robot gets feedback from the user by detecting human gestures (blue arrow) and interpreting it.
*Human gets feedback from the robot by interpreting gestures performed by the robot (green arrow).

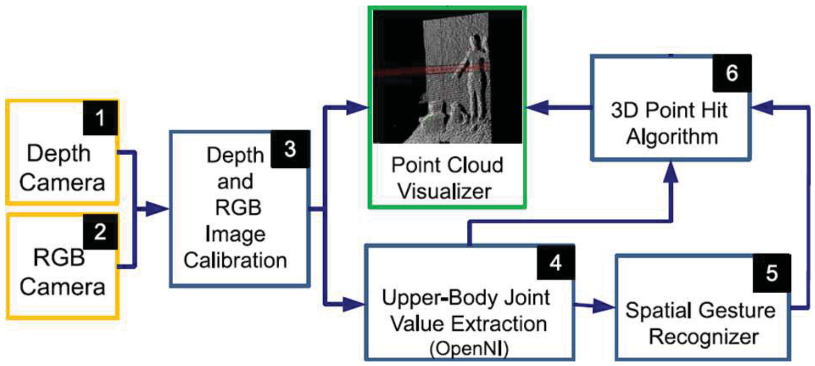

1
Depth Camera
2
RGB Camera
Depth and RGB Image Calibration
3
Point Cloud Visualizer
3D Point Hit Algorithm
6
Upper-Body Joint Value Extraction (OpenNI)
4
Spatial Gesture Recognizer
5

Figure 4.6: Selection block diagram: Input from the depth and RGB sensor (1,2,3) is used to identify human joint positions(4). Based on the joint positions pointing direction and location is infer(5,6)

interaction scheme, we use three states. In the first state, the user stands in front of the Kinect and makes a neutral calibration pose. This allows the NITE skeletal tracking library to initialize by doing background substraction, human silhouette identification, and skeleton joint-value extraction. The transition from first state (calibration/initialization) to the second state (target selection) is done when the user points to the target with her dominant hand and raises her non-dominant hand over her shoulder to confirm the pointing direction. If the user lowers her non-dominant hand below her shoulder, the state shifts to the third state, in which the hit location is calculated.

For building the hit algorithm, we used two pointing direction configurations, the line between head and dominant hand (LHH) and line between dominant elbow and dominant hand (LEH). Having the pointing direction, we need a 3D world representation for finding the 3D hit point. We used the PCL library [86] for extracting and manipulating the 3D point cloud from the depth camera sensor. Then, we add the location of the user torso joint values inside the point cloud, which allows us to find the pointing location looking at the LHH or LEH configuration. After acquiring the pointing direction by having either LHH or LEH configuration, we can define a parameterized line equation:

$$\vec{\mathbf{l}} = \vec{\mathbf{J}}_{upper} + t\,(\vec{\mathbf{J}}_{hand} - \vec{\mathbf{J}}_{upper}), \tag{4.1}$$

where $\vec{\mathbf{l}}$ is the parameterized line vector, $t \in \mathbb{R}^+$ is a non-negative real-valued step of the parameterized line, $\vec{\mathbf{J}}_{hand}$ is the 3-vector of the 3D hand coordinates, and $\vec{\mathbf{J}}_{upper}$ is the 3-vector of the 3D elbow or head coordinates, depending on the chosen configuration. For our ray-to-point-cloud searching algorithm, we can simply replace the parameterized line inside the sphere equation, returning a discrete set of spheres located across the pointing line:

$$\| \vec{\mathbf{x}} - (\vec{\mathbf{J}}_{upper} + t(\vec{\mathbf{J}}_{hand} - \vec{\mathbf{J}}_{upper})) \|^2 = r^2, \tag{4.2}$$

The point cloud generated by the depth sensor is structured in an octree and the algorithm iterates over the multiple spheres defined in (4.2) for a possible hit point. Fig 4.7 shows a user inside the point cloud using the LHH direction configuration where the red sphere is the tracked position of his head, the blue sphere his hand and the green sphere the target location.

4.3.1.2 Gesture Identification

Based on the Pick-and-place task steps described in Section 4.2 we define four human body gestures for the human-robot interaction, **Stand by** (Figure 4.4B), **Pointing** (Figure 4.4A), **Yes** (Figure 4.4D) and **No** (Figure 4.4C). Our Gesture Identification block is in charge of interpreting the predefined human gestures. We use the OpenNI skeleton tracking libraries [88] to find human joint locations. Gesture identification is based on spatial relations between the different human joint locations, *e.g.,* the **No** gesture is detected when the dominant hand of the user is inside a sphere with a 30 cm radius and center in the non dominant shoulder. Notice that the gestures are selected for the particular pick-and-place task example application, but other body gestures can be easily defined for other type of interactions.

4.3.1.3 Object Localization

This block receives data from the depth camera as input, and outputs the centroid locations and bounding box of one or more objects located on the plane of interest (Figure 4.7C). We use the point cloud library [86] to manipulate data coming from the depth sensor. Below are listed the steps that we follow:

- Downsampling: In order to achieve real-time performance in our system, we start by downsampling the point cloud obtained from the Kinect, with a voxelized grid approach. We also filter non-relevant data.
- Horizontal planar segmentation: Using RANSAC [91] we find the table plane coefficients and inliers.
- Convex hull 2D: Using the inliers and coefficients from the above step, we find the points that belong to the table.
- Extract inliers: We remove every point in the point cloud except for the points inside the volume defined by the table plane and a maximum object height.
- Euclidean cluster: We cluster the inliers by distance to distinguish the objects.
- Centroids calculation and bounding boxes: We calculate the mean vector for each cluster and the smallest bounding box.

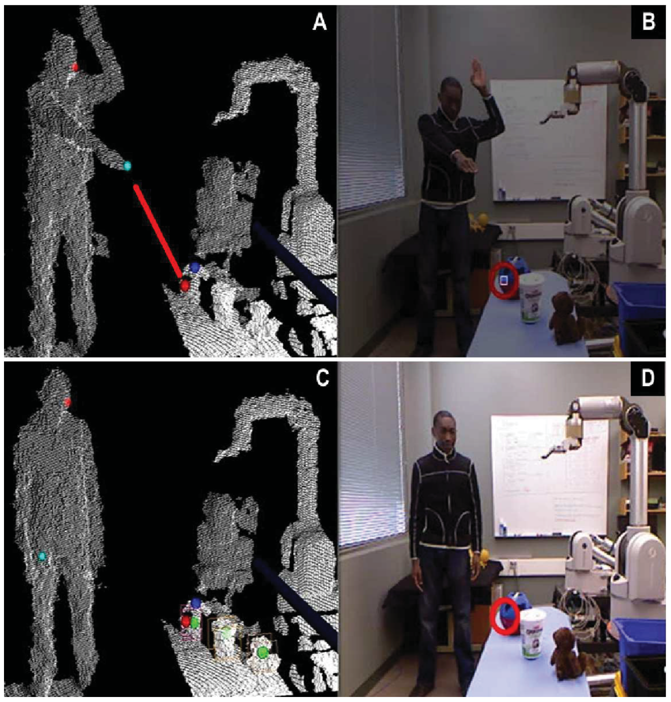


Figure 4.7: System point cloud visualization (A,C) and RGB visualization (B,D): Human Selecting object by pointing (top). Red and light blue spheres attached to the human head and hand respectively (A). These two locations are used to find a virtual 3D ray (see 3D red ray between the hand and the hit point, A). Centroids (green spheres) and bounding boxes extraction from objects over the table plane (C). The system corrects the interpreted hit point (red sphere) to the closest object (shape puzzle toy). The dark blue sphere above the selected object represents the correction done by the system and is used as the motion target for the robot finger when doing the confirmation gesture.

4.3.1.4 Decision

The Decision block provides our system with the capacity of interaction. This block outputs the robot interaction and receives as input the hit point location from the Selection block, the identified gesture preformed by the human from the Gesture Identification block and the object centroids and bounding boxes from the Object Localization block (See Figure 4.5).

The Decision block is based on the state machine shown in Figure 4.8, which consists of six states:

- *State 1, Initial State:* The human is using the **Stand by** gesture. The system

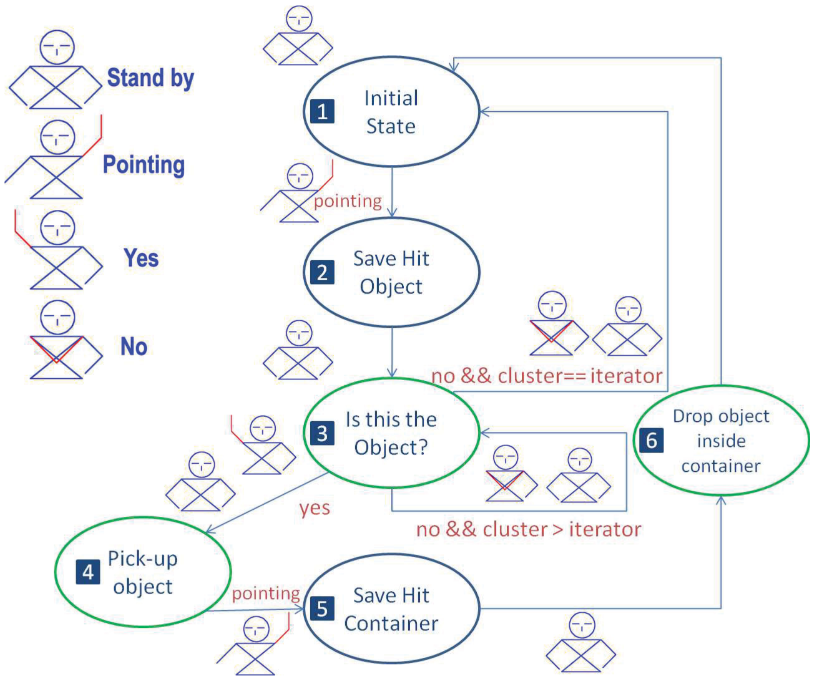


Figure 4.8: Finite State Diagram. The system is activated when the user selects an object by pointing. Six states are needed for the complete pick-and-place interaction.

holds until a **Pointing** gesture is performed by the human.

- *State 2, Save Hit Object:* The system saves the hit 3D point coordinates coming from the Selection block. The system holds until the human goes back to the **Stand by** gesture.

- *State 3, Is this the Object?:* Using the input data from the Object Localization and Selection blocks, the system calculates the nearest object centroid to the hit point. Then, the system projects the centroid position into the top face of the object bounding box (dark blue sphere in Figure 4.7). The robot, uses the centroid projection location to position the robot end effector over the selected object and gives feedback to the human performing the pointing gesture (see Figure 4.4B). The system holds until the human performs a **Yes** or **No** gesture. If the gesture is **No** followed by a **Stand by** gesture, the system iterates to the next nearest object and reinitializes state 3. In the case where the human keeps doing the **No** gesture until there are no objects left the robot goes back to state 1. If the the human performs a **Yes** gesture followed by a **Stand by** gesture the state is shifted to state number 4.

- *State 4, Pick-up object:* Using the selected object centroid projection (dark blue sphere in Figure 4.7), the robot positions its hand above the projected object centroid with its palm perpendicular to the table plane and grasps the object. Then the robot goes to an initial position and waits for a **Pointing** gesture indicating the dropping location.

- *State 5, Save Hit Location:* After the **Pointing** gesture is performed the hit target is saved and compared with the possible available locations. The system chooses the object closest to the hit point.

- *State 6, Drop object inside container:* The robot places the object by dropping it inside the selected container and goes back to the initial position, State 1.

Notice that the decision block is tailored in its designed to the pick-and-place example application. However, our system is based on a state machine representation and as shown by [92] a spanning variety of desirable behaviours for home robots can be easily configured with such a system.

4.3.2 Robot-Human system

In the Robot-Human case the gesture interpretation is done by the human (see Figure 4.2B), which makes the system implementation simpler than in the human-robot case. We implement **robot-Pointing** (Figure 4.2B) and three symbolic gestures for the robot: **robot-Stand-by**, **robot-Yes** (top Figure 4.9): The robot moves its right finger up and down repeatedly, **robot-No** (bottom Figure 4.9): The robot moves its wrist from right to left repeatedly. Similar to the robot-human, symbolic gestures can be replaced by pre-recorded voice feedback. In contrast to our human-robot configuration, where objects are dynamically detected, we used predefined positions in our implementation for simplicity. We pre-record the robot pointing gesture directions for the different object locations in the two test configurations, objects on a line and normal (objects spread out over the table). We interact with the user by predefined robot configurations.

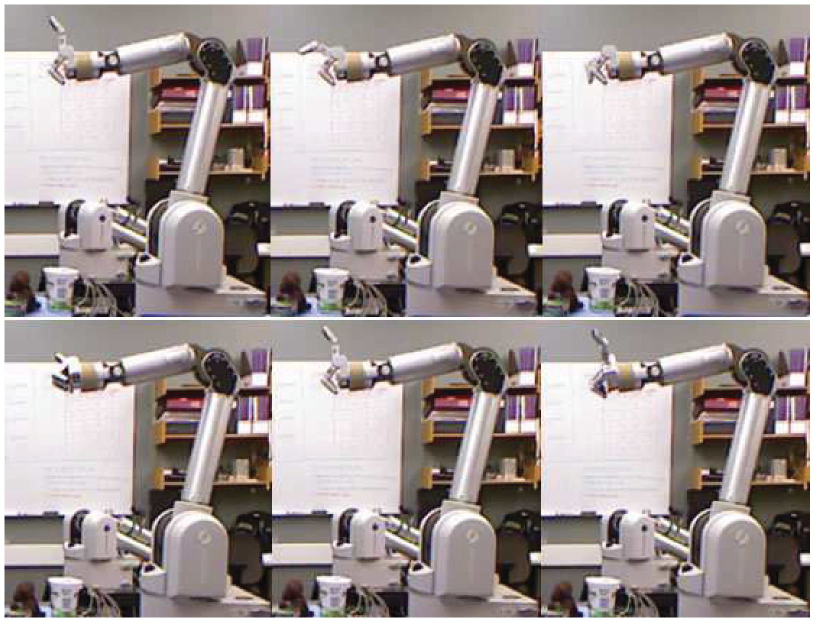


Figure 4.9: Top: Robot performing a "robot-Yes" gesture. The robot moves its right finger up and down repeatedly. Bottom: Robot performing a "robot-No" gesture. The robot moves its wrist from right to left repeatedly while holding its right finger up.

4.4 Experiments and Analysis

We first ran experiments for studying human pointing accuracy and precision interpretation. Then we performed three experiments using the instructor-assistant pick-and-place task described in Section 4.3.

4.4.1 Pointing accuracy and Precision

In this section we quantify both human and our robot system pointing accuracy and precision interpretation. Here, accuracy is defined as the mean value of the sample data whereas precision is the standard deviation *i.e.,* accuracy is the Euclidean distance between interpreted point and the target location, while precision is the uncertainty of the interpreted point.

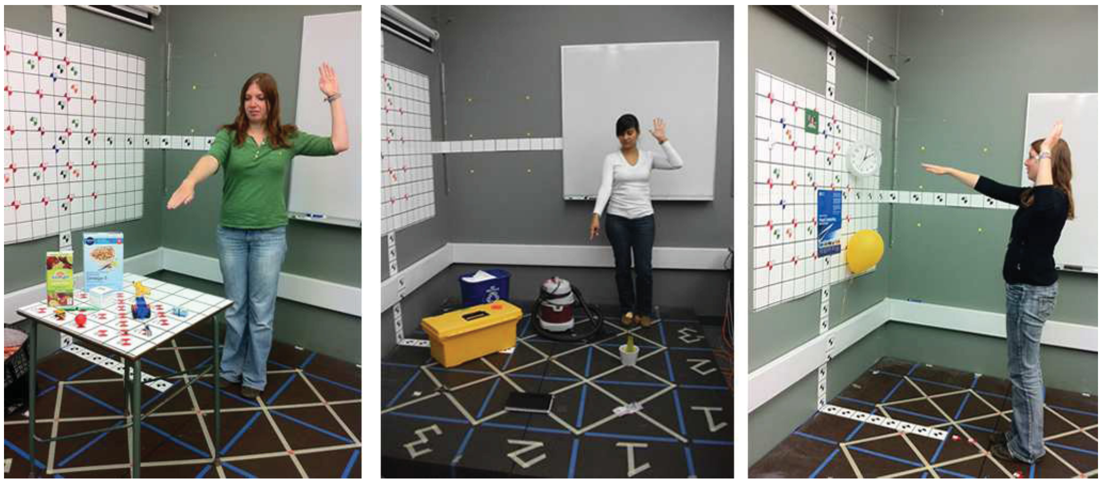


Figure 4.10: Experiments, Left:table, Middle:floor, Right:wall

4.4.1.1 Experiment Setup

We divided the accuracy and precision experiments into three main sections: "table", "floor" and "wall". Each section had a duration of approximately 20 minutes per participant. For the human interpretation experiments, we had 8 participants, five male and three female. All them with normal or corrected to normal vision, seven right-handed and one left-handed. Their ages varied between 18 and 33. For the system interpretation experiments we had 9 subjects, seven male and two female, all with normal or corrected to normal vision and right-handed. Their ages varied between 18 and 34.

All three main sections were divided as follows:

- For the "table" section (left Figure 4.10) a tabletop (height 67 cm) was divided into a grid of 5 by 6 squares with 10 cm side length each. In the middle of these squares, numbered targets were attached (red targets Figure 4.12a). For pointing on the table, the subject stood with a distance of approximately 0.5 m on one side of the table.
- For the "floor" section (middle Figure 4.10) a plateau was divided into 3 by 3 squares of 50 cm and marked with numbers for rows and columns(Figure 4.12b). The subject was asked to stand at different locations for pointing at particular intersections or objects.

- For the “wall” section (right Figure 4.10) a poster of 14 by 10 squares of 10 cm side length each was attached and equipped with numbered targets (Figure 4.12c). In addition, there were a horizontal (right) and a vertical row (up and down) of squares attached to the poster to measure the accuracy over a bigger area. The subjects had a distance of 80 cm to the wall centred in front of the poster (right Figure 4.10).

4.4.1.2 Human pointing interpretation experiment

The eight participants were grouped in pairs; a list of 37 random coordinates for each scenario was given to participant number one. He was asked to point to the specific coordinates on the list. The second participant was asked to interpret the pointed coordinates and write them down. Then, the roles were exchanged and a new random list was given to participant number two. No talking or other type of communication were permitted between participants. The results of pointing accuracy and precision interpretation are shown in Figure 4.11. On average the human interpretation accuracy for the three scenarios was 16.6 cm.

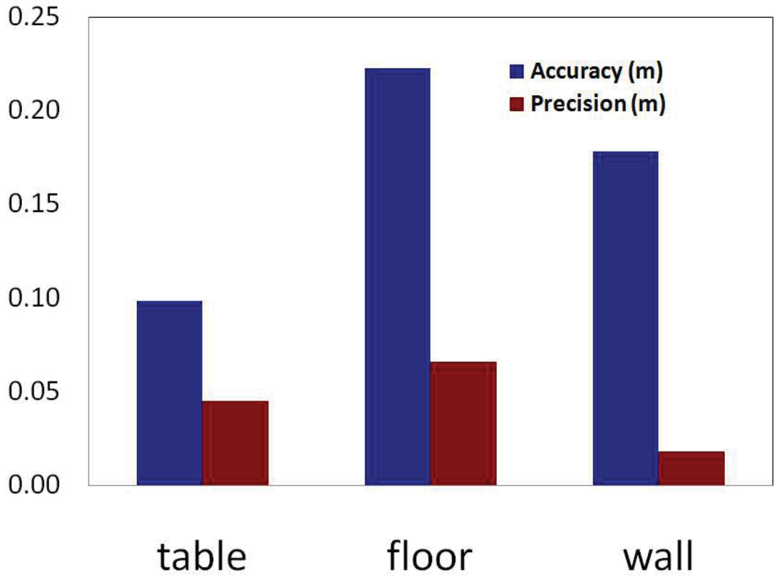


Figure 4.11: Human experimental pointing accuracy and precision interpretation

4.4.1.3 System pointing interpretation experiment

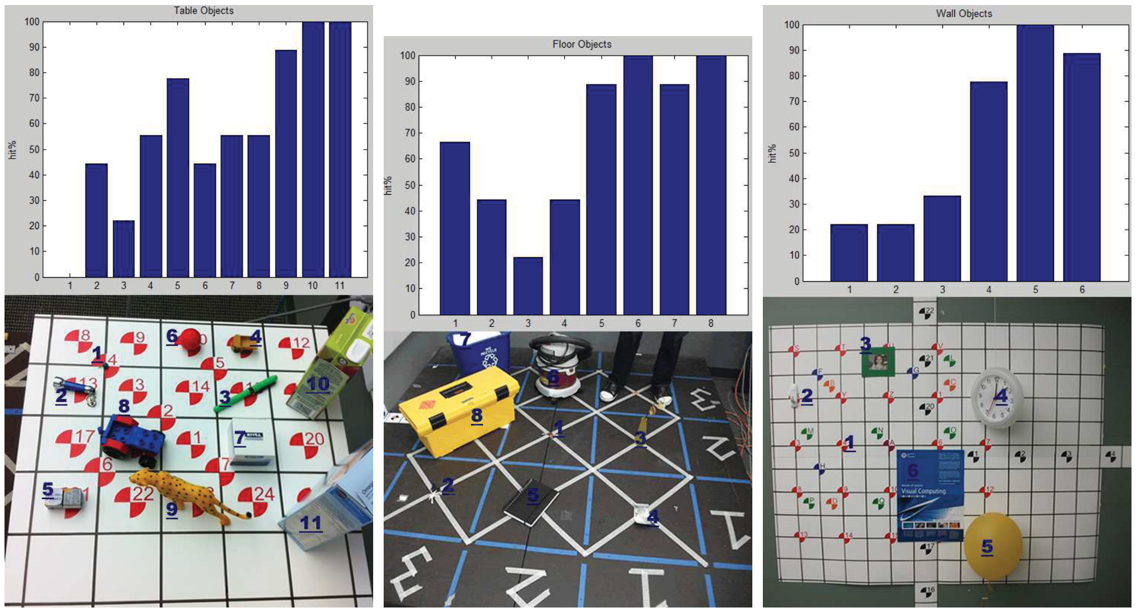
All set-ups consisted of a Microsoft Kinect simultaneously running the RGB and depth camera on a regular Linux machine with our SEPO module. Before the pointing process, each subject had to go through the calibration process of standing facing the Kinect, putting up both hands, waving them back and forth and taking a step towards or away from the Kinect until they were successfully tracked. After that, the subject was asked to go to the pointing position. The person was allowed to orient herself freely and suitably for the pointing direction. Sections "table" and "wall" were subdivided into 11 short tasks, and section "floor" into 10. For all the sections, in the first two tasks, the subject was asked to point at the called out number (bottom Figure 4.12a) or intersection (bottom Figure 4.12b) and to select the pointing position by raising the left arm.

No further instructions were given. The subject was supposed to point as naturally as possible. While carrying out the first task, the LHH configuration was applied and in the second task, the LEH configuration. In the third and fourth tasks, the subject was told that the system is running with the LHH strategy now. In the fifth and sixth tasks, the participant was asked to point with the LEH strategy. The accuracies of those tasks with the knowledge of the strategies were compared and the following tasks were carried out with the more accurate strategy. In the last task (eleventh for "table" and "wall", tenth for the "floor"), real objects were added to the scene and the subject was asked to point at them.

After the last experiment, the subjects were asked to complete a questionnaire. The evaluation showed that the subjects did not think the accuracy of their pointing changed in the three scenarios. The self-reported overall difficulty was rated in the middle of the spectrum (variance 5.61) and overall fatigue slightly under medium (3 of 7 with 7 as very high, variance 4.44). As the variance in both of these ratings shows people had a different sense of how difficult and tiring the pointing and selecting was. The head and hand alignment strategy was preferred, with only two participants preferring to point with the forearm. We also asked how annoying the gesture of raising the hand for finally selecting the object was. The participants rated the annoyance in the middle of the spectrum with a variance of 2.78.

The experimental results are summarized in Figure 4.13.

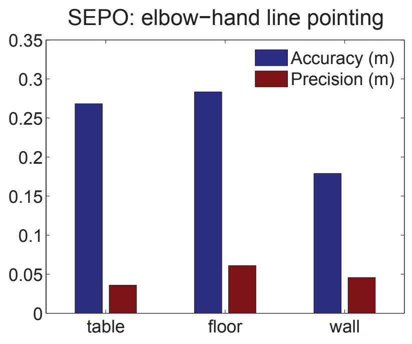
Our results confirm published data which says that pointing is more accurate and precise in the case where the pointing direction is computed from the head-hand line



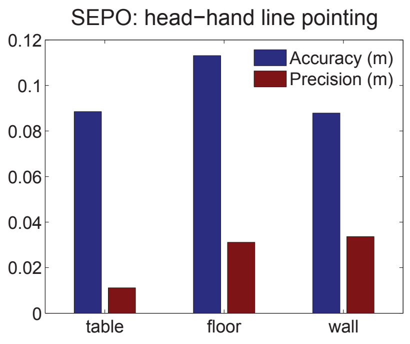
(a) (b) (c)

Figure 4.12: (a) Hit percentage in the "table" scenario, numbers in blue in the bottom image correspond to numbers in the bar-graph. For instance, when users point to the juice box they were 100% accurate. However while pointing to the ping-pong ball hit% was 45. (b) Hit percentage in the "floor" scenario, numbers in blue in the bottom image corresponds to numbers in the bar-graph. For instance, when users point to the tool box they were 100% accurate. However during pointing to the bill hit% was 43. (c)Hit percentage in the "wall" scenario, numbers in blue in the bottom image corresponds to numbers in the bar-graph. For instance, when users point to the balloon they were 100% accurate. However, while pointing to the picture frame, hit% were 33.

than in the case where it is computed from the elbow-hand line [93]. In addition, our results suggest head-hand pointing on the table with SEPO module is more precise than pointing on the wall and the floor. This can be explained by fact that the table set-up presents an adequate ratio of the proximity of the subjects to the visually tracked area of the objects. The accuracies of pointing to the wall and on the table are respectively 2.5 cm and 2.4 cm better than for the floor. After task 6, the following tasks were carried out with the more accurate strategy which was LHH, experiments are summarized in Figure 4.14 with 5 of our participants. For the table scenario, the subjects point twice repeatedly to four different locations on a numbered grid (see the picture of the table in Figure 4.12a). The results indicate an average accuracy of 8.8 cm across all participants within a [5.4, 12.4] cm range. To give a task related sense

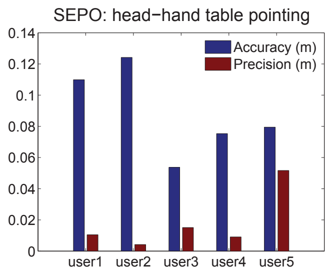


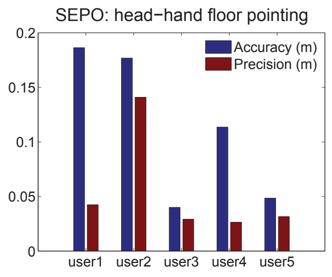
(a)



(b)

Figure 4.13: Elbow and head hand-line pointing accuracy and precision in three scenarios: "table", "floor", and "wall". Notice that the LHH configuration is more precise than the LEH configuration





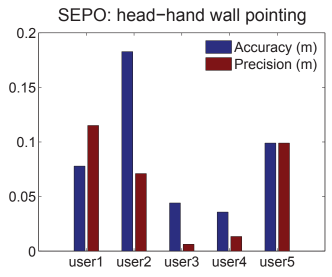


Figure 4.14: Different users accuracy and precision results for LHH configuration

of the accuracy, we can say that on average users would have no trouble pointing to bigger objects (*e.g.,* cereal box, juice box); see bottom Figure 4.12a). The average precision acquired for this experiment was 1.1 cm within a $[0.4, 5.2]$ cm range. For the floor scenario, the subjects repeatedly pointed 4 times to 3 different locations on the grid floor (see Figure 4.12b). The results indicated an average accuracy of 11.3 cm across all participants within a [4.0, 18.6] cm range. The average precision acquired for this experiment was 3.1 cm within a $[2.7, 14.1]$ cm range, users would have no trouble pointing to objects like toolbox, garbage bin, vacuum cleaner(see Figure 4.12b). For the wall scenario, the subjects point twice to each of 3 different locations on a grid pattern on the wall (see Figure 4.12c). The results indicate an average accuracy of 8.8 cm across all participants within a [3.6, 18.3] cm range. The average precision acquired for this experiment was 3.4 cm within a [0.6, 11.5] cm range, users would have no trouble pointing objects like poster, balloon, wall-clock (see Figure 4.12c).

4.4.1.4 Evaluation of 3D Point Hit Algorithm on Everyday Targets

To validate our experimental results, we performed a visual evaluation using our system's visual output (Figure 4.7). The experiment included real objects for each of the scenarios. Through the visual system output, we detected if the object was successfully selected. If the object was missed, we noted how many squares it was away from the object. The average hit percentage was 58% for all the objects on the table, 100% for big objects (10,11 in Fig. 4.12a), 57% for medium objects (7,8,9 in Fig. 4.12a) and 41% for small objects (1,2,3,4,5,6 in Fig. 4.12a). The average hit percentage was 69% for all objects on the floor, 96% for big objects (6,7,8 in Fig. 4.12b), 52% for medium objects (3,4,5 in Fig. 4.12b) and 55% for small objects (1,2 in Fig. 4.12b). The average hit percentage was 51% for all objects on the wall, 89% for big objects (4,5,6 in Fig. 4.12c), 33% for medium objects (3 in Fig. 4.12c) and 22% for small objects (1,2 in Fig. 4.12c). The difference in hit% between the scenarios is due to the relation between surface area projected in the user's field of view and the distance to the target object. Bigger objects (e. g. table: cereal box, juice box;floor:vacuum cleaner, tool box;wall:balloon, poster) were hit more often than smaller ones (picture frame, flower pot or pin). When an object was missed, the hit point was almost all the time in an area of one square (10 cm on the wall, 50 cm on the floor) from the desired object. The results are summarized in Figures 4.12a,4.12b,4.12c).

4.4.2 Human and Robot Bi-directional gesturing

After studying precision and accuracy of both human and system, we performed the instructor-assistant pick-and-place task example application described in Section 4.3, where the actors for each experiment are: human-human, human-robot and robot-human. We had a total of 8 participants aged from 18 to 34 with 6 male and 2 female. Among them, 5 had corrected vision and one was left handed. The average time per participant to complete the three experiments was 1 hour including break times.

The different experimental set-ups are shown in Figure 7.1. Two arrangements of objects were used: normal and line. In the normal configuration objects are spread over the 2D table surface (Left Figure 4.15), while in the line configurations, objects are collinear with the instructor (Right Figure 4.15). In the second case only the arm tilt angle is informative, while in the first, general case, both tilt and pan angles help indicate what object the instructor is pointing to.

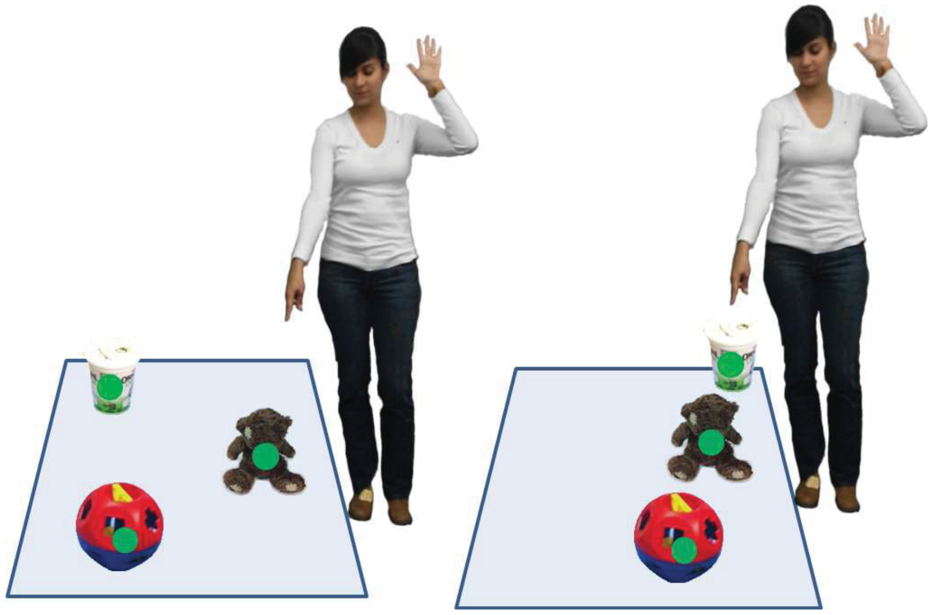


Figure 4.15: Object Arrengements.
Left: Normal configuration. Easy to infer which object she is pointing (bear).
Right: Line configuration. It is hard to infer which object she is pointing to from this point of view (yogurt, bear, shape puzzle toy).

For both human-human and human-robot scenarios, the instructor and the assistant visual sensors (eyes location for human and kinect location for the robot) are 2.70 m apart from each other. In addition, the table is positioned between both of them. More precisely the table (0.67 m height for the human-human configuration and 0.87 m height for the robot-human configuration) is set at a distance of 1.08 m (1.00 m in front of and 0.40 m to the left of the instructor) from the instructor in order to avoid directly touching of the objects while pointing at them. For the robot-human setup, the human was positioned perpendicular to the kinect sensor's field of view (Figure 7.1B), repectively 1m and 1.8m from the intersection point where the table was located. For both robot-human and human-robot experiments, the 7DOF WAM arm was located such that the table space belonged to the robot arm workspace (see Figure 7.1B and Figure 7.1C).

4.4.3 Human-human

The human-human case (Figure 7.1A) took an average of 3.05 minutes per participant. In this case, we performed 4 experiments with 10 instructor-assistant pairs made from 8 participants. In the first experiment, we considered a normal configuration (*e.g.,* square configuration) of 4 objects on the table (see Figure 4.4B). In the last 3 experiments, we considered differently ordered line arrangement of the 4 objects (see Figure 7.1A and Figure 4.7B). The linear arrangement is purposely made to be aligned with the pointing direction of the instructor. The motivation of using this configuration is to test experimentally if a human is inferring a pointing direction, does point of view matters? In particular, an aligned configuration (Right, Figure 4.15) is more difficult to infer than a normal configuration (Left, Figure 4.15). The last experiment was the longest (1.1 min) since participants needed more time to use the pre-determined gestures (Section 4.3.1.2) which had been introduced to them at this stage.

Experimental results are shown in Figure 4.17. In the human-human case success ratio for the line configuration is 0.75 while for the general configuration is 0.95. It is clear then that it is harder to interpret the pointing in the former case without any feedback.

4.4.4 Robot-human

The robot-human case (Figure 4.4) took an average of 8.15 min. Here we performed a set of 3 experiments with 8 participants. We considered both the general arrangement configuration and the line arrangement configuration of the objects. Results are shown in Figure 4.17 where we can see that the general object configuration without feedback is equally difficult as the line configuration without feedback. We believe that the difference in results obtained between the human-human case and the robot-human is explained in the under actuation nature of our Barrett hand, which makes it difficult to interpret pointing location.

4.4.5 Human-Robot

The human-robot case (Figure 4.4C) took an average of 2 min. In the human-robot interaction case the success ratio is equal in both the general configuration and the line configuration see Figure 4.17. This means that our vision system can better interpret human pointing than a human assistant.

Figure 4.16 shows that the number of misinterpretations of the assistant is lowest in the human-robot interaction. In fact there is 28% misinterpretations in a robot-human interaction, 10% misinterpretation in a human-human interaction and 2%

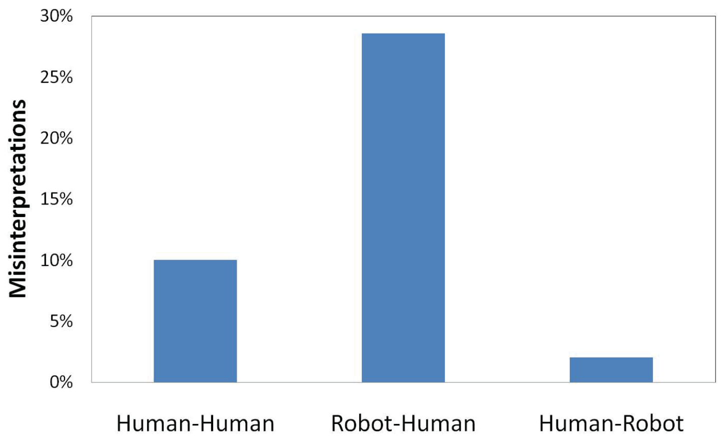


Figure 4.16: Feedback misinterpretations

misinterpretation in a human-robot interaction. The high percentage of misinterpretations in the first case can be explained by the fact that our Barrett WAM Hand pointing finger could not be made straight. This is a technical limitation with this hand because of the under-actuation constraints used to build it. This suggest that robot designers that are planing to integrate human-like gestures during robot interaction should take into consideration not only the mechanical capacity of the robotic part but also its shape and human visual interpretation during operation.

After finishing the experiments, participants were asked to complete a questionnaire. On average participants found it equally difficult to accomplish the task with feedback in the human-human, robot-human and human-robot interactions. In the robot-human case, it is an obvious result due to the anthropomorphic shape of the robot. In the human-robot case, the result shows that our system interprets pointing and gives feed-back at least as well as humans do. Figure 4.21 shows the questionnaire results. A Likert-type scale from 1(easy) to 7(difficult) was used. Human-human symbolic gesture interpretation was found less difficult with average score of 1.8. However, interpreting symbolic gestures performed by the robot got a similar result, average score: 2.2, for instance, using **NO** and **YES**, robot symbolic gesture were easily understood by humans. Using predetermined gestures to interact with the robot was scored with 2.5. This result may increase if the number of predetermined gestures is increased; notice that all predefined symbolic gestures can be replaced by predefined voice commands. Users scored "pointing to the right object" (see Figure 4.21) on average with 3. The main difficulty being distinguishing objects close to each other.

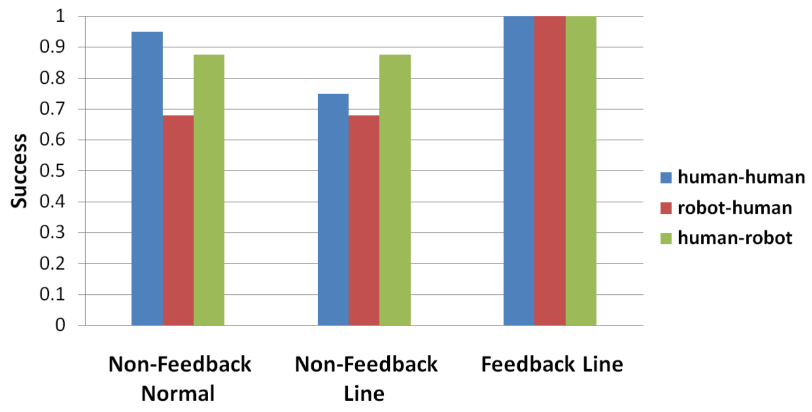


Figure 4.17: Instructor and assistant experiment success rate

Interpreting both human pointing and robot pointing was the most difficult, with a score of 4. To sum up, our experiment results have validated that pointing interpretation is harder for humans than for our system in specific cases of object arrangement. This system is thus suitable for a human-robot non-verbal interaction with pointing gestures for pick-and-place tasks since pointing interpretation is more precise than human-human interaction.

4.4.6 Other gestures used in human-human interaction

During the human-human experiments we told the subjects to non-verbally communicate what items they wanted the other person to pick up, but we did not tell them what specific gestures to use. This allowed us to observe what gestures the subjects chose. Most of the subjects simply pointed towards the target object; however we also saw some other variations. A couple of subjects used their hands to indicate the shape of the desired objects. That is, they formed a round shape to show they wanted the soccer ball. Similarly, another subject used this same technique to tell the other person to pick up the yogurt container. Another gesture we observed was a kind of counting gesture. When the objects were ordered in a line, the subject moved his/her forearm in a circular fashion three times to indicate he/she wanted the third object. The forearm was held parallel to the front of his/her body. We also observed another technique to distinguish between objects at different distance from the subject. Two subjects stood on their toes while pointing toward target objects that were farther away. In particular this was also used to distinguish between the blue and the black containers. Finally one of the subjects used curved pointing when the objects were placed in a line. That is, instead of pointing directly towards the object in a straight line, the subject formed a curve with his/her arm to allow for more precise selection. These observations gave us new ideas for improving our current pre-determined gestures.

4.5 Application Example: "Making Pizza with my robot"

We envision that our system can be used in different real life scenarios e.g., a robot can work behind a counter taking the role of a shopkeeper; a client points to a particular object and by using confirmation feedback the robot will reach the desired product. In another situation a robot can be used as a chef at a hotel breakfast buffet; the client points to different ingredients to include in his omelette. In a metal workshop a robot can assist a welder by picking and placing parts. The welder only has to point to them, avoiding heavy weight manipulation and extreme temperatures.

To bring our study to a practical situation we made our robot capable of preparing pizza by gesturing with a human. The application set-up is shown in Figure 5.2. Ingredients are arbitrarily placed on top of the table and detected. When the user gets close to the cooking table the robot is activated and the human tracking starts, see Figure 4.19. The user then can select any ingredient by simply pointing to it. After the selection the robot picks the ingredient and pours it on top of the pizza tray, Figure 4.20 A-C. This action can be repeated as many times as the user wants. After the user is satisfied with the number and amount of ingredients, a "finish pizza" gesture is performed and the robot places the pizza inside the oven, Figure 4.20 D-F. A video of the complete interaction can be found at `http://webdocs.cs.ualberta.ca/%7Evis/HRI/makingPizza.wmv`.

The example application demonstrates that a complex task for a robot, like preparing a customized pizza for a client, can be simplified by using the appropriate communication interface. We strongly believe that by researching simple and novel ways of human robot communication will make robotic applications more common in human environments.

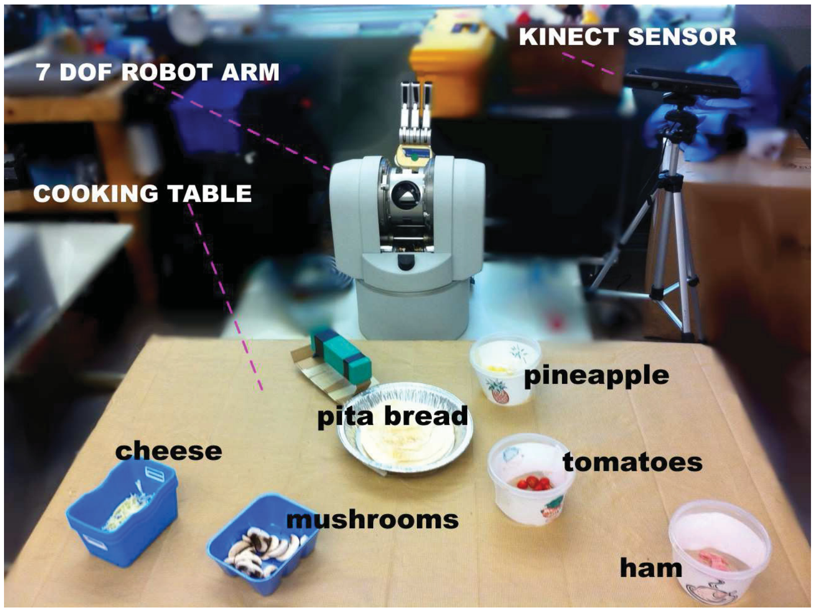


Figure 4.18: The set-up used in our practical application “making pizza with my robot” consists of a 7DOF robot arm, a kinect sensor and a cooking table with ingredients.

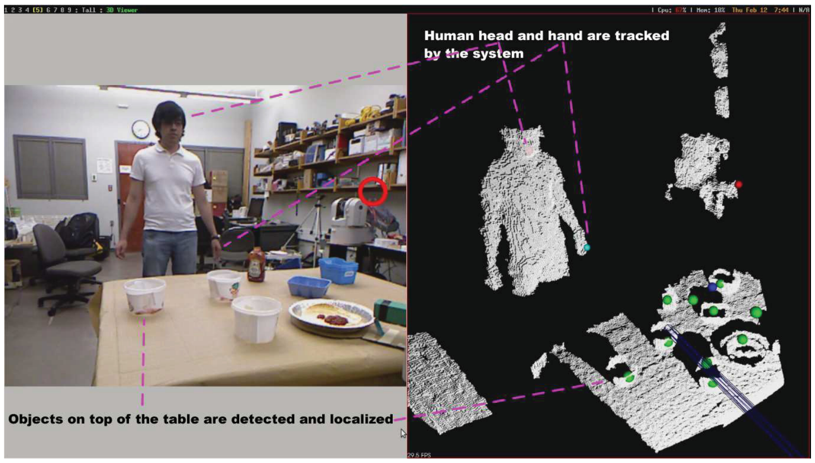

Human head and hand are tracked by the system
Objects on top of the table are detected and localized

Figure 4.19: Left: RGB visualization. Right: Object detection inside the point cloud visualization.

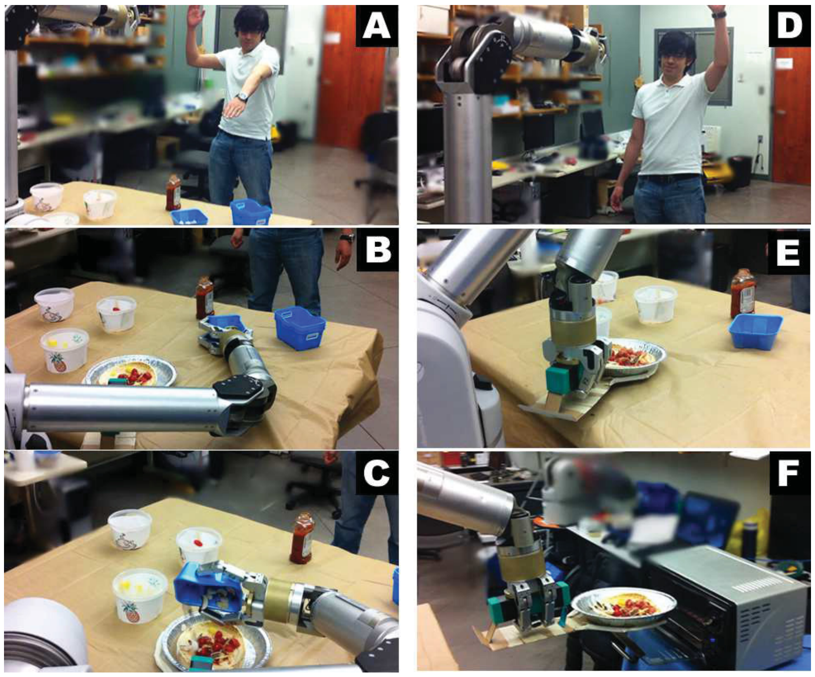


Figure 4.20: Left Column: User selects mushrooms by pointing, the robot picks and pours mushrooms in the pizza tray. Right Column: User performs the “finish pizza gesture” and the robot places the pizza inside the toaster oven.

4.6 Conclusions

When humans collaborate on manipulation tasks, gestures form an integral part of the communication. It is often easier to point to an object or desired location than to describe a position in words or numbers, robots can benefit from this. We designed and evaluated through experiments a robot and vision system that is able to see, interpret and act using a gesture based language. We started by studying human pointing in three specific household scenarios: “table”, “floor”, and “wall”. Human accuracy interpretation in average was 16.6$\pm$2.4 cm. In our proposed system we tested two different pointing configurations *line between head and dominand hand* (LHH) and *line between dominant elbow and dominant hand* (LEH). Our results have demonstrated that the LHH configuration consistently outperforms the accuracy and precision of the LEH configuration. Furthermore through experimentation we have shown that our system interpretation is 1.7 more accurate than human interpretation in same conditions. Our system enables users to point to and select objects with an average position accuracy of 9.6$\pm$1.6 cm in household situations, that means we can successfully select objects similar to cereal box on the kitchen table, a notebook on the desk, a poster on the wall, a vacuum cleaner on the floor, etc. Through our robot vision system, we mitigate pointing imprecision by adding to our robot the ability to interpret which object the user is pointing to and corroborate the interpretation by gesture based feed-back. In our experimental study, 8 humans interacted with the robot for about 1h each. We showed experimentally that our system can behave similar to, and in same cases better than, a human interpreting human pointing. The task was to clean off a table and sort the objects into containers. We compared human-robot, robot-human and human-human pairs as instructors and assistants respectively. We performed the tasks, both with instruction only (one way communication) and with feedback gestures from the assistant (robot or human) to verify that robot/human had interpreted the task correctly. Without feedback, the assistant could interpret the pointing gesture correctly in 70-95% of the cases. Humans had particular difficulty at distinguishing objects placed on a line (75% success rate), but were much better with the objects in a general configuration (95% success). Humans also had more difficulty interpreting the robot's pointing (70%) than other human pointing. This is likely to the physical inability of our Barrett robot hand to extend the finger fully and point with a straight gesture towards the object. However such limitation is common for robot hands. The robot vision system had similar accuracy independent of object configuration (88% success). In the feedback case, the assistant indicated the selected object by pointing just above it. The instructor could then confirm with a yes gesture, or deny with a no gesture, and then point again to the desired object. This feedback allowed successful task completion in all cases for both the robot and human assistants. In questionnaire answers, the human subjects indicated that, for this task, the human-robot system was about equally easy to work with compared to human-human communication.

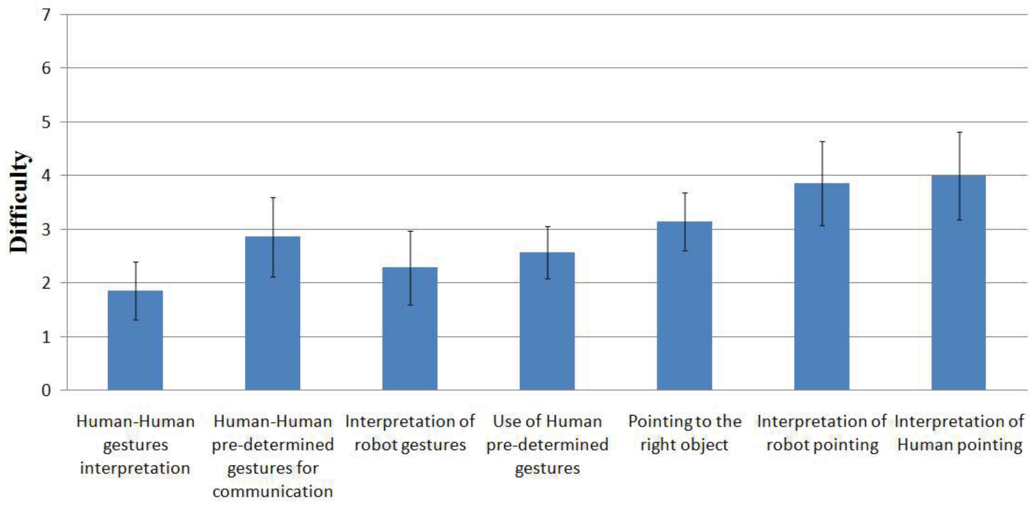


Figure 4.21: Data collected by questionnaire

Chapter 5

Proximate Display-based pointing for upper body-disabled persons

Wheelchair mounted robot arms help humans with upper extremity disabilities such as muscular dystrophy or spasms to perform daily tasks. While relatively recent, such robot arms and hands are used by hundreds of disabled users in Europe and North America. Usually arm and hand motion are directly commanded using a regular 2DOF motorized wheelchair joystick, but this is difficult and tiring, especially for disabled users.

Our hypothesis is that current joystick control solutions can be dramatically improved by designing a intuitive user-oriented interface capable of reducing the interface input DOFs, cognitive load and task accomplish time. We present and experimentally evaluate a human-robot interface, where the detailed joystick command motions needed to reach for objects is replaced by computer vision object detection and motion guidance. A robot camera records the scene and displays on-line the image with detected objects highlighted to the user. The user can select objects and desired grasp orientations interacting with the screen interface. In addition we design and integrate a vision force module that provides the user with compliant pulling/pushing and rotating capabilities. We prototyped and evaluated several varieties of image-based object and grasp selection interfaces under guidance from rehab experts and disabled users. We do not aim to replace user control, but instead to augment user capabilities through our system with different levels of semi-autonomy, while leaving the user with a sense that he/she is in control of the task.

We have implemented a portable ROS-based system and demonstrated on both a Barrett WAM arm and a Kinova Jaco arm. Experimentally we validate the usability in unstructured pick and place tasks by performing an orientation and drinking task on a group of 8 control subjects and one pilot user. The study showed that overall our system was about 70% faster compared to joystick control and on a standard questionnaire subjects indicated on average 4 times lower cognitive load.

5.1 Introduction

Robot arm and hand manipulation can help the upper body disabled with everyday manipulation tasks, just like power wheelchairs provide mobility. While power wheelchairs are already common, assistive robot arms and manipulation is an emerging area that has the potential to help a similar number of people as wheelchairs have. In recent years assistive robot arms have become light and relatively affordable. However these arms still rely on joystick commands, which are slow and difficult to use. While many associate robotics with autonomous operation, the number of tasks that can be solved autonomously in a regular, unstructured household environment is very small [14]. Semi-autonomy is more promising, in that it can relieve the human of the detailed control of the many motion degrees of freedom of a robot arm and hand, without restricting the set of tasks that can be performed.

A key challenge is to develop and validate natural and efficient Human-Robot Interactions (HRI). State of the art rehabilitation robot arms try to duplicate human arm functionality by providing 6 or 7 degrees of freedom (DOF) motion. Typically manufacturers provide a joystick interface or a customized input device based on the end user's disabilities.

Typical user groups of robot arms suffer from disabilities such as spasms or muscular dystrophy [94]. Mapping from the high DOF arm to a 2DOF joystick requires switching between modes which couple the joystick to different translation, rotation and grasp motions. This mode switching is time consuming, cumbersome and increases complexity and cognitive load for the users. For instance [95] presents a usability study on the JACO arm [22] where they quantified the number of mode switches required to complete common daily living tasks, e.g., eating soup took in average 70 mode switches and opening a door 18 mode switches.

While tele-manipulation is the norm in rehabilitation, much of the robotics research focuses on autonomy. User interaction can be minimized by ceding more autonomy to the robot. However, as pointed out by Kim *et al.* [96], disabled users preferred to

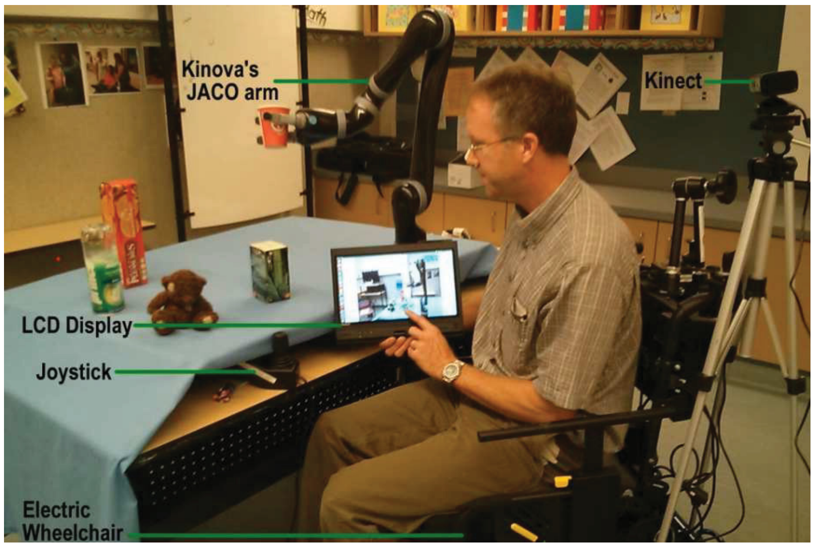


Figure 5.1: Vision-based User Interface to a 6DOF robot arm and hand. By pointing and selecting in a 2D video image of the scene, the user can point to objects, select grasp types and execute robot actions. The system interprets the 2D pointing and selections with respect to the 3D scene geometry and objects, and plans the 6DOF motion trajectories. Thus the system relieves the user of the difficulty of the direct control of a high DOF robot arm and hand system.

keep as much control as possible with the aim of reasserting their domain of interaction with their environment as well as to engage and exercise their cognitive abilities to the fullest.

We develop a computer vision-based system that focuses on user interaction. Through our system we augment user capabilities with an interface that offers different levels of autonomy, that can be easily adapted to different degrees of disability, for object reaching and grasping. This is achieved while leaving the user with a sense that he/she is in control of the task.

The main strengths in our system compared to the others described in section 2.1.5 are adaptability and simplicity. Our interface turns the complexity of a multiple-DOF robot arm into a 2D object selection interface. Although our system is capable of doing autonomous grasping, our aim is to enhance user control rather than replace it.

The rest of this chapter is organized as follows: Section 5.2 describes our system, we first describe the system operation and functionalities, and then we provide technical detail of the implementation. Section 5.3 shows our experimental set-up and results. Discussion and conclusions are present in sections 5.4 and 5.5, respec-

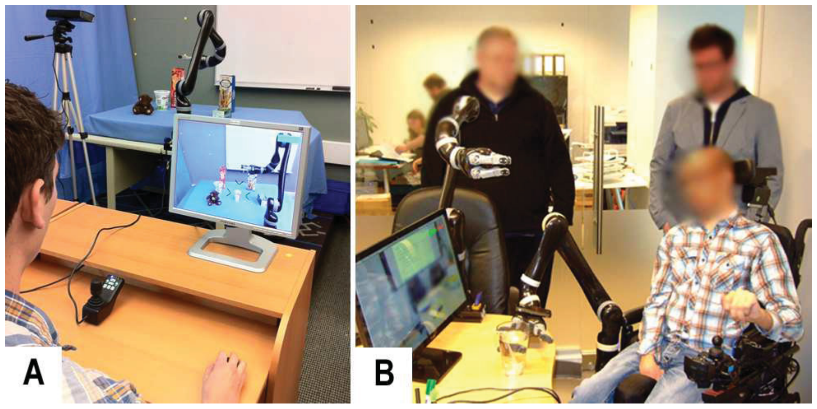


Figure 5.2: Vision-Based Interface. **A.** Experimental set-up for control group. **B.** Disabled pilot user 1 testing our interface.

tively. This chapter is a significant extension of our earlier work [97] with addition of compliant modules and more numerically detailed experiments, observations and improvements.

5.2 System Description

Our proposed system uses a robot-arm, Kinect sensor, screen and a low power computer. Figure 5.2 illustrates our test set-ups using the Kinova's JACO arm [22]. Typically, electric wheelchair users equipped with a robot-manipulator use the same wheelchair motion control device to interface with the robot-manipulator. The input device is adapted to the capabilities of the user by a physical therapist who selects an interface device appropriately for the handicap. These are usually especially adapted joysticks. These input devices are often already designed to allow the user to operate a PC through a USB port. We tap into this mouse emulation function for our interface.

In our system, the scene in front of the user is shown on a screen, and the user can then select an object (see Figure 5.1). We have had the chance to discuss, test and refine our interface with a disabled end user of our system (pilot user 1), and handicap robot engineers at the robot company Kinova. We performed three iterations of three interfaces with feedback from Kinova. Given their feedback a minimalistic interface mode was set as default, and the two more general modes are fall-backs when the minimalistic system does not solve the task. The main feedback received during our visits includes:

- Users prefer a system that works reliably with manual intervention to a system that is autonomous but not reliable.
- Automating the 6DOF translation and rotation needed to reach and align for a grasp is a high value goal.
- Users like to have direct control rather than autonomy when they perform a fine manipulation *e.g.,* grasp, poke object, drink. This is consistent with the study performed by [96].

5.2.1 Levels of Autonomy in the User Interface

As mentioned previously our aim is not to replace user control. We instead augment user capabilities through our system with different levels of semi-autonomy, while leaving the user with a sense that he/she is in control of the task. We propose three types of object selection: **1D List Selection**, **2D Image Plane Selection**, **General Pick Selection** and one placing mode: **General Place**. In addition we address physical environment interaction: pulling/pushing and rotating tasks through the **Compliant task Selection**.

5.2.1.1 1D List Selection

After interacting with our first pilot user who has spasms, this control mode was designed for upper body disabled with high mobility restrictions. Discrete selection of the target object from a list only requires a trigger signal from the disabled person. In our testing with pilot user 1 we threshold joystick movement magnitude (in any direction) to step though the object list and grasp modes (see Figure 5.3). Notice that the grasping cursor provides the user with a way of grasping in cluttered environments e.g., in Figure 5.3C the user selects the right approach instead of the left to avoid hitting the blue container with the robot end-effector.

5.2.1.2 2D Image Plane Selection

This selection mode is intended for users with an ability to efficiently control the joystick coarsely in 2D. Instead of linearly iterating through the list of detected objects, the user clicks on the 2D image with his handicap mouse emulation interface. The system then highlights the detected object closest to the click (using the 3D point cloud) and selects it (Figure 5.7A). After selecting the object, the grasping cursor

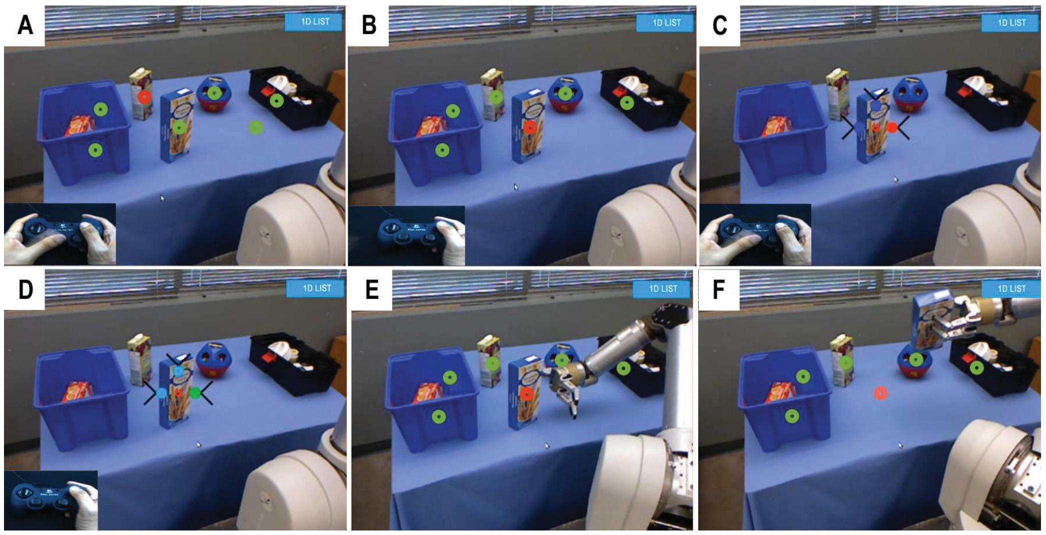


Figure 5.3: 1D List Selection: Here motion commands to the robot are given by discrete selection from displayed options.
A. Computer vision detects objects on horizontal planes (tables, counters), generates a linear list of the objects and highlights the detected objects with green circles in the user image.
B. The user iterates through the list until the desired object is marked with a red circle. For users with spasms we threshold joystick vibration to trigger iteration.
C. A grasping cursor appears. The user can iterate again over the three possible gasping approaches: right, left and top.
D. After selection the grasping cursor, lights, green or red indicate if the grasping is possible or not.
E. The robot moves close to the selected object with the desired grasping orientation.
F. User can retake control of the robot and finish the grasping by joystick teleoperation or command the system to finish autonomously.

appears and by placing the mouse cursor over one of the blue dots of the grasping cursor, the system displays a green or red dot indicating if the grasping location can or cannot be achieved.

5.2.1.3 General Pick Selection

When the system is not able to detect an object in the scene correctly, or simply if the user wants a more flexible way to interact, **Pick Selection** allows an approximate positioning of the robot end-effector to any desired 3D location by having the user click in a 2D image and compute the corresponding 3D location. Figure 5.4A shows the **Pick Selection** mode. In the **1D List Selection** or **2D Image Plane Selection**, the system considers information from the detected object (see section 5.2.2.3), in the **General Pick Selection** however the only information is the selected point and the desired orientation. Thus, a fixed offset is set to maintain the robot end-effector above the table's plane.

5.2.1.4 General Place Mode

After picking up an object, the user can select a scene location for placing it, *e.g.,* in Figure 5.4C the user wants to place a box of juice inside the blue container. In this case, the user clicks inside the blue container, and then selects a grasping approach by clicking in the grasp cursor. The robot-arm moves to an approximate location slightly above the desired location, with the selected grasping orientation. The user then finishes the task via teleoperation. In drinking and other proximal tasks it is common in assistive robotics to have pre-recorded poses. The user can also bring the object to any of these.

5.2.1.5 Compliant task Selection

Pick and place objects are the most frequent daily activities performed by a person [98] and it is the main functionality presented in our interface. Other common activities like opening a fridge, drawer, container, turning a door knob, etc. can also be generalized as an exposed functionality in our interface.
A common factor in these tasks is the contact with the environment, which needs to be handled compliantly. To the best of our knowledge, only a few works consider compliant features in their assistive robotic systems [94]. Handling this in the right way ensure the safety of the user and extends the equipment life cycle. By using the compliant task cursor (See Figure 5.5) the user first selects the target object and then decides between pulling/pushing or rotating operations. After the object selection the system calculates the 3D location and the direction of the motion. Two detail examples of both rotating and pulling/pushing tasks are illustrated in Figure 5.5.

Notice that the novelty of our interface relies on the simplicity of its visualization. Even though the point cloud output is available, we present only a simple 2D RGB image, to the user; where we hide the complexity of the 3D grasp orientations by

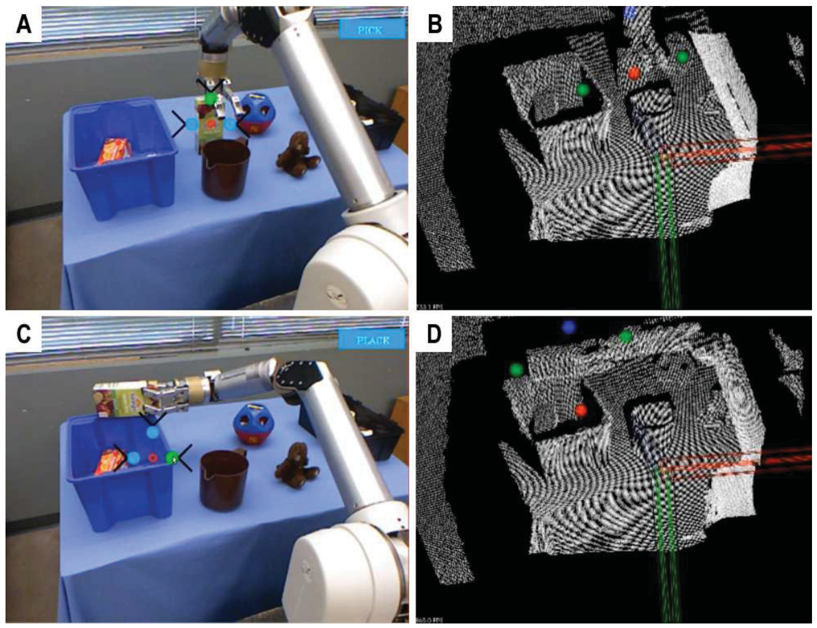


Figure 5.4: Left column shows our system user interface, right column shows our system point cloud visualization.
A. User's view. The Pick mode is activated. The grasping cursor was used to position the robot from a top grasping approach.
B. Point Cloud view. The red sphere corresponds to the 3D location of the selected point in the RGB image. The blue sphere to the top grasping location, and the green spheres the right and left grasping locations.
C. The Place mode is activated. The grasping cursor is used this time to place an object. A right placing location is selected.
D. The red sphere corresponds to the 3D location of the select point in the RGB image. The blue sphere to the top placing location and the green spheres to the right, left placing locations. Notice that the only difference between Pick and Place modes is the distance between the green, blue and red spheres.

discretizing them.

5.2.2 System

Our system is composed of four modules as shown in Figure 5.6: **Interface**, **Vision**, **Robot-Arm** and **Calibration** modules. A detailed description of each module is given below.

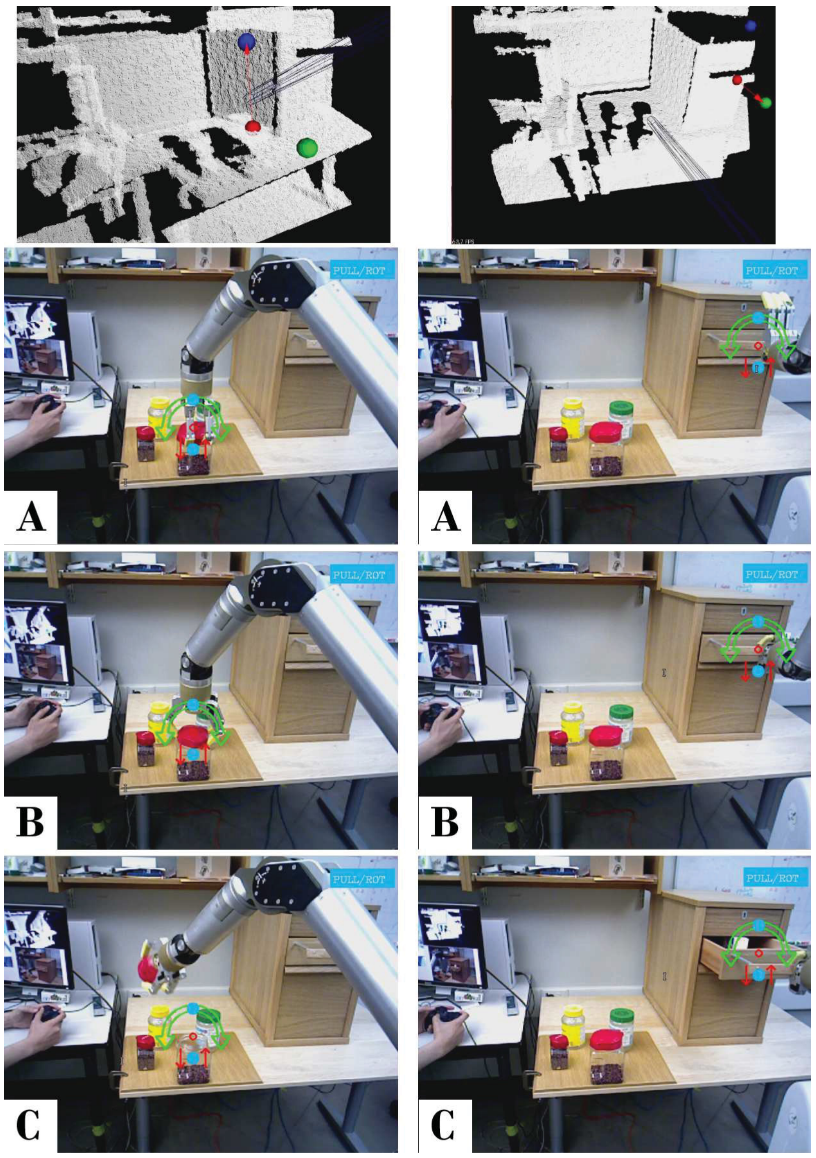


Figure 5.5: Left: Opening a jar task: A) After the user selects the container lid, the system calculates the 3D location and the normal direction of the lid surface (see the top-left point cloud image). Based on this information, the robot's end-effector is aligned facing the lid. B) If required, the user performs the fine alignment of the robot (through the gamepad), and then activate the rotation module which allows the user to control the rotation of the lid in a compliant mode. C) This action could be repeated several times till the lid is completely loose.
Right: Opening a drawer task: A) After the user selects the drawer handle, the system calculates the 3D location and the moving direction of the handle surface (see the top-right point cloud image). Based on this, the robot's end-effector is aligned facing the drawer. B) If required, the user performs the fine alignment (through the gamepad), and then activate the pulling/pushing module to complete the task.

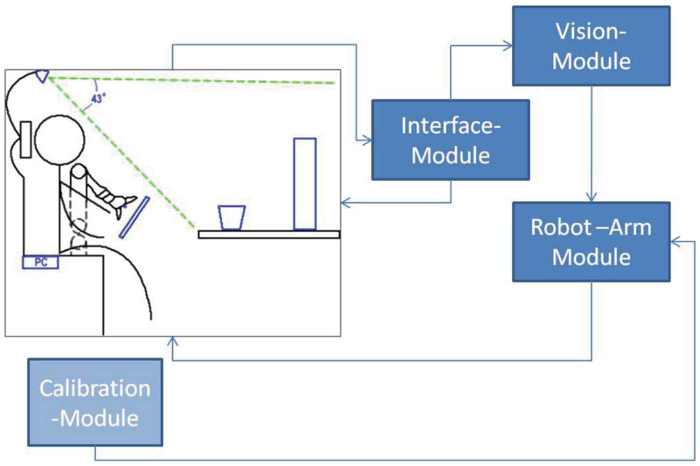


Figure 5.6: System Diagram. Our proposed system uses a Kinect camera, screen and regular Linux PC. The Vision and Robot-Arm modules abstract the complexity of the arm which is presented to the end-user through the Interface module as a 2D image interface.

5.2.2.1 Interface

The person views the regular 2D RGB video from the Kinect scene camera on a screen. All user interaction is defined with reference to this visual interface. A blue label in the top right of the interface indicates the current mode (Figures 5.4 A and C) and by clicking on it, the modes can be cycled, in the case of the **1D List Selection**, the blue label (mode changing) can be included in the iterative selection to avoid clicking.

5.2.2.2 Calibration

Our vision-based system assumes that the RGB-D (*i.e.,* kinect) camera is fixed on the same reference frame of the robot-arm (*i.e.,* the arm and the Kinect are mounted in the same wheelchair). We designed a visual 2D interface for calibration purposes. The arm moves the end-effector through a set of predefined points $R_i(x, y, z)$ corresponding to motions along the robot-arm frame of reference. The user then clicks on a known marked point on the robot at each of the i locations on the 2D image. Using the depth data from the Kinect the corresponding 3D points $K_i(x, y, z)$ are found. At the end of the calibration process the transformation matrix relating the Kinect's frame of reference and the robot's frame of reference is stored and used in our main application. For practical purposes, this calibration will not change unless the Kinect camera is moved to a different place in the wheelchair. The complete calibration process requires the user to select the marked point 6 times and takes approximately 3 minutes. We estimated experimentally the precision of our interface is 1.78 ± 0.67 cm, when the user clicks in a point in the image that belongs to the robot workspace.

5.2.2.3 Vision

This module provides all the vision functionality using the kinect camera, which is composed of a depth and RGB camera.

For the **General Pick** and **General Place** modes (see Section 5.2.1.3 and 5.2.1.4) the 3D coordinates associated with the selected pixel in the RGB camera are found using the depth and RGB camera correspondence. The point cloud representations of the **General Pick** and **General Place** modes are shown in Figure 5.4B and D respectively. The red sphere represents the 3D location of the selected point in the RGB image. The green spheres are the left and right grasping locations and the blue sphere the top grasp location. The hand orientation is calculated based on the grasp location and the red sphere. The distance between the grasping points and the selected 3D location is fixed, meaning that the user has to be careful while using this mode because, if the object selected is big enough to be inside the grasping points then the end-effector can collide with the object. When using this mode, the system provides a step trajectory mode (see Section 5.2.2.4) which allows the user to move step by step through the arm trajectory in such way that the user can stop the movement before any collision.

The **1D List Selection** and **2D Image Plane Selection** use a more elaborate approach. First the objects in the nearest horizontal plane are detected. The *Point Cloud Library* [99] is used to manipulate our data coming from the depth sensor. Below we list the steps that are followed to detect the objects in the horizontal plane closest to the Kinect:

- *Downsampling:* In order to achieve real-time performance in our system, we start by downsampling the point cloud obtained from the Kinect with a voxelized grid approach. We also filter non-relevant data.

- *Horizontal planar segmentation:* Using RANSAC [91] we find the table plane coefficients and inliers.

- *Convex hull 2D:* Using the inliers and coefficients from the above step, we find the points that belong to the table.
- *Extract inliers:* We remove every point in the point cloud except for the points inside the volume defined by the table plane and a maximum object height threshold.
- *Euclidean cluster:* We cluster the inliers by distance to distinguish the objects.
- *Centroids calculation and bounding boxes:* We calculate the mean vector for each cluster and the minimum bounding box.

After having the object's centroid and estimated bounding box, the grasp locations are calculated for the top left and right as follows. The top grasp (blue sphere in Figure 5.7B) is calculated as a factor of the object bounding box height.

For calculating the right and left grasp points, a grasping ring around the selected object (purple spheres in Figure 5.7B) parallel to the plane that holds it, is defined. From the RANSAC procedure [91], we have the plane coefficients and from there one can easily calculate a normal vector $\mathbf{n} = \{n_x, n_y, n_z\}$ to the plane. Next the vector $\mathbf{a} = \{1, 1, \frac{-n_x - n_y}{n_z}\}$ parallel to the plane is found. Having $\mathbf{n}$ and $\hat{\mathbf{a}} = \{a_1, a_2, a_3\}$, where $\hat{\mathbf{a}}$ is the normalized vector of $\mathbf{a}$, we find $\mathbf{b} = \hat{\mathbf{a}} \times \mathbf{n} = \{b_1, b_2, b_3\}$. Based on these vectors, the location of the grasping spheres around the object centroid is defined. The following formulation will hold for any plane inclination with respect to the robot.

$$\mathbf{X}(\Theta) = \mathbf{c} + rcos(\Theta)\hat{\mathbf{a}} + rsin(\Theta)\mathbf{b}$$

The radius r of the ring is calculated based on the width value of the object's bounding box. The angular separation between spheres is Θ and the centroid of the object $\mathbf{c} = \{c_x, c_y, c_z\}$. The point cloud visualization of our system is shown in Figure 5.7B. The toy bear was selected, using the **2D Image Plane Selection** where the blue sphere and purple spheres around the toy bear were constructed as described above.

For the **Compliant task Selection**, once the object is selected, the vision module feeds the *Robot Arm* submodules: *pulling/pushing* and *rotating* module with the 3D coordinates of the object and the corresponding surface normal. The surface normal is calculated by fitting a plane to the neighborhood P_i of the target point p_i. P_i is either formed by the k nearest neighbors of p_i or by points within a radius r from p_i. Given P_i, its covariance matrix $\boldsymbol{C}_i \in R^{3x3}$ is computed (see [82] for

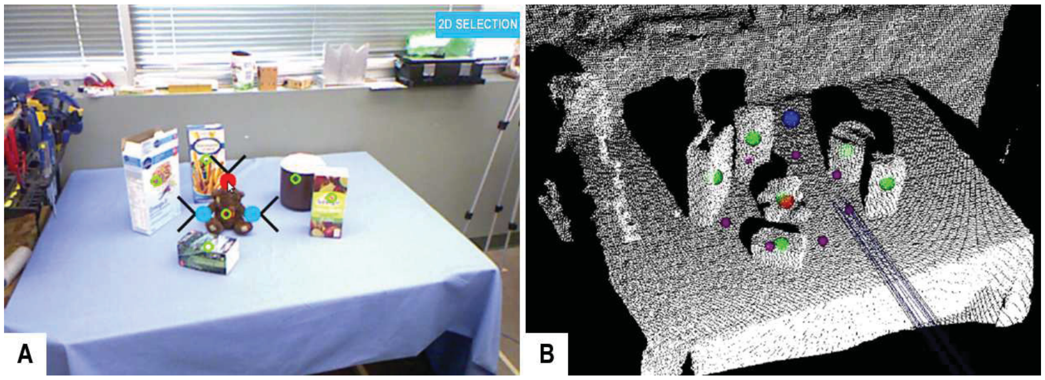


Figure 5.7: 2D Image Plane Selection.
A.The 2D Image Plane Selection is activated (blue label top right). Objects detected in the scene are marked by a green ring. The grasping cursor is being used to position the robot from a top grasping approach.
B.The green spheres are the 3D centroid locations of the objects detected in the scene. After selection is done, a red sphere appears representing the 3D location of the selected point in the RGB image. The system corrects the selection to the nearest centroid and constructs a ring of purple spheres used for the right and left grasping. A blue sphere is also added for the top grasping.

details). The eigenvectors of the covariance matrix, $\boldsymbol{v}_{i,0}$, corresponding to the smallest eigenvalue $\lambda_{i,0}$, can be used as an estimate of the target normal $\hat{\boldsymbol{n}}_i$ [100]. To reduce the computational time and achieve real time performance, we used the integral normal estimation implemented in the *Point Cloud Library* (PCL) [99]. This implementation takes advantage of the organized structure of the point cloud acquired by the RGB-D sensor and also uses a pixel neighborhood instead of a spatial neighborhood. By using the pixel neighborhood the simplest approach is to use the right-left pixels and up-down pixels to form two local tangential vectors and calculate the normal using the cross product. However, since the data is noisy due to the nature of the sensor, an average tangential vector calculation (known as a integral image [101]) is performed. In our implementation we used the PCL *Average 3D Gradient* mode which creates 6 integral images to compute smoothed versions of horizontal and vertical 3D gradients and computes the normals using the cross-product between these two gradients [102].

5.2.2.4 Robot Arm

We use ROS and Moveit!, [103], [104]. This allows for rapid deployment across different arms and enables us to leverage work done by the robotics community. In order to add support for new arms within the system, we require a model of the arm and a simple interface to extract or set angular joint positions of the arm. In the case of the Jaco arm we later removed the ROS dependency and utilized the built in Cartesian planner to reduce system requirements. Through this module, arm trajectories are generated given the grasp location and orientation calculated by the *vision* module. Additionally, it is possible to define constraints on regions within the arm's workspace. We create a collision object aligned with the nearest plane found by the *vision* module in order to avoid collisions with the table surface on which the objects rest. Once a trajectory is generated, it is then interpreted by the arm's interface and the appropriate commands to modify the arm's joint angles are sent. The trajectory can be executed completely autonomously, or iterated manually by the user. The latter allows for finer control, giving the user the option to stop the execution of a generated trajectory and continue the task manually.

Rotating module This module computes the required torque to rotate the end-effector in the desired direction compliantly from the following control law:

$$\boldsymbol{\tau}_t = \boldsymbol{K}_{r,1}(\boldsymbol{q}_d - \boldsymbol{q}) + \boldsymbol{K}_{r,2}(-\dot{\boldsymbol{q}})$$

where $\boldsymbol{q}_d, \boldsymbol{q}$ are quaternions representing the desired and actual orientation of the robot's end-effector. Here, the desired orientation $\boldsymbol{q}_d$ is computed from the normal direction received from the vision module. The commanded joint torques to the robot is $\boldsymbol{\tau} = \boldsymbol{J}^T\boldsymbol{\tau}_t$, where $\boldsymbol{J}$ is the task Jacobian matrix of the robot.

Pulling/Pushing module This module computes the required forces to move the end-effector in the desired linear direction compliantly from the following control law:

$$\boldsymbol{F} = \boldsymbol{K}_{p,1}(\boldsymbol{v}_d - \dot{\boldsymbol{x}}) + \boldsymbol{K}_{p,2}(-\ddot{\boldsymbol{x}})$$

where $\boldsymbol{F}$ is the force required to move the robot in the desired direction $\boldsymbol{v}_d$ with a constant velocity. Again, $\boldsymbol{v}_d$ is obtained from the normal direction received from the vision module. The actual commanded torque to the robot is $\boldsymbol{\tau} = \boldsymbol{J}^T\boldsymbol{F}$.

5.3 Experiments

Our first pilot user was involved during our interface design along with rehab experts. He suffers from cerebral palsy. Due to his condition, he has restricted upper limb movements and suffers from spasms. Based on his feedback and Kinova's rehab engineers, we performed some modifications to our original system, the most relevant being: (1) instead of using a left grasping we do a front grasping. Our pilot user 1 at Kinova commented that in his daily use of the robot arm, a front grasping is more common than a left grasp. Furthermore, due to the particular right handed configuration of the JACO arm, which differs from the WAM which can be used as a left or right arm, doing a left grasping with the JACO arm is unusual. (2) Although MoveIt! brings a lot of functionalities to our system, it is computationally expensive which, for a lab setting, is not a problem, but for daily use in a power wheelchair, is not desirable. In our collaboration with Kinova to port our system to Jaco we remove MoveIt! from our portable implementation and used the manufacturer API, this guarantees low power consumption and real time execution. Our second pilot user has a form of quadriplegia. He has no hand or finger movements, just gross arm movements. He performed user studies along with an 8 subject control group to evaluate our interface. Our control group consists of 8 participants with normal hand-eye coordination aged from 18 to 34 with 6 males and 2 females. Among them, 7 had corrected vision. We performed two experiments for testing our system: an *orientation task* (Figure 5.8) and a *drinking task* (Figure 5.11). Both the pilot study user and our control group ran the two experiments. The average time per participant to complete the two experiments was 1 hour including break times and an initial 10 minute training period. Demonstration of some of the experiments can be seen in the supplementary videos [105].

5.3.1 Experimental setup

The operator station consists of a mouse, screen and joystick. In front of the control station, a Kinect camera was located facing a 100x160cm tabletop with the JACO arm attached to the table (see Figure 5.2A). We adapt a joystick to the limitations of our pilot user and integrate a mouse pad that he uses on daily basis for internet navigation (see Figure 5.9).

5.3.2 System accuracy

The accuracy of our system depends on the robot and Kinect sensor accuracy and the calibration between both. Our experimental setup to calculate the system accuracy consist in the robot and our calibration module running on it and a grid pattern with

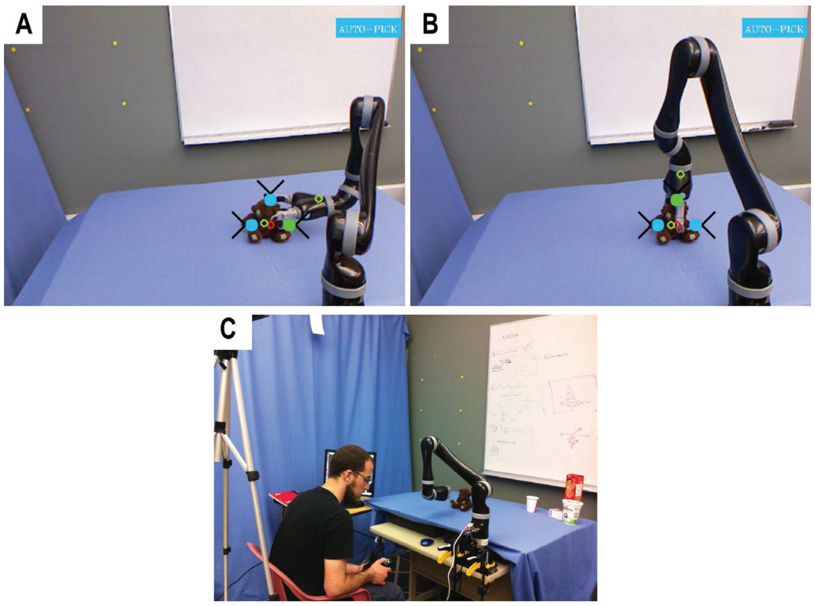


Figure 5.8: Orientation Task. User was asked to locate the robot hand in a pre-grasp position for three specific orientations right (A), top (B) and front (C).

1cm divisions place on the table workspace. We placed a marker on top of the grid and using our interface we select a position in the grid. After the robot moves to the selected location we physical measures the error between the robot end-effector position and the marker location. We perform this several times in different locations in the workspace, the average error is 1.78 ± 0.67 cm.

5.3.3 Tasks and data Analysis

5.3.3.1 Orientation task

The first experiment is a pre-grasp orientation task. An object was placed over tabletop and the user was asked to locate the robot hand in a right, top and front orientation with respect to the object (see Figure 5.8). Two control methods were used: (1) Direct teleoperation through JACO's built in Cartesian controller driven by a joystick and (2) Visual interface selection mode. To avoid bias, order of selection of the control method and orientation order was randomized for each user. During the task, the user was asked to complete 9 grasp orientations. During the experiment, we record the time it takes the user to complete each orientation and the number of times the user switches modes with the joystick to achieve translation, rotations and grasp motions of the arm. A comparison between the average time to complete the

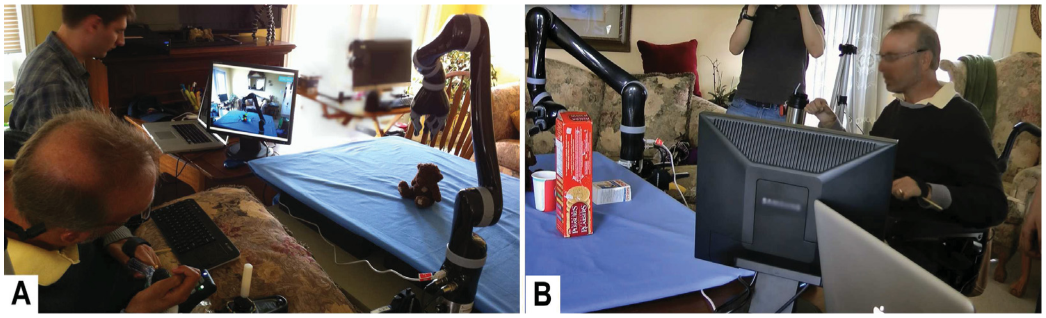


Figure 5.9: Set-up for our pilot study. For direct teleoperation, we adapted a joystick. For interacting with the visual interface, we integrated end-user's regular PC mouse-pad. (A) Orientation task, (B) Drinking task.

different orientations using the joystick and our proposed visual interface is shown in Figure 5.10. In the three orientation cases, the visual interface outperforms the joystick, for both the pilot study and the control group. The easiest orientation was right followed by front and top. This was expected because the JACO arm was mounted as a right handed one, where its starting position is a right orientation.

For the control group, a paired t test for comparing the two modes for each orientation was performed. We wanted to know if the interface decreases operation time as compared to the joystick mode, *i.e.,*$H_0 : \mu_d = 0$ versus $H_a : \mu_d > 0$ with a significant level $\alpha = 0.05$. Here $\mu_d = \mu_{Joystick} - \mu_{Interface}$ where $\mu_{Joystick}$ and $\mu_{Interface}$ are the mean time to complete the orientation using the joystick and using the interface respectively. The t and P_{values} for each orientation are shown in table 5.1. Since $P_{value} \leq \alpha$ we reject H_0 : for all the orientations. This data analysis confirms that the interface mode decrease operation time in comparison with the joystick mode. A paired t confidence interval is also computed. Pairing the samples, the interval is given by $\bar{x}_d \pm (tCriticalValue) * \frac{S_d}{\sqrt{n}}$, where S_d is the standard deviation and n is the number of samples. Using a 95% confidence interval, the last column in table 5.1 shows the average range improvement in seconds between the joystick teleoperation and our interface (*i.e.,*we can be 95% confident that grasping from the top with our interface saves between 8.74 to 34.05 seconds in comparison with the joystick teleoperation). On average, our interface was faster for positioning and orienting the robot than the joystick interface. Disabled people in general have more difficulty providing input as seen in the joystick teleoperation performed by the pilot user, as shown figure 5.10. However since our visual user interface reduces the amount of input compared 6DOF direct robot control, the same task time can be reduced in

Orientation	t	P-value	Range
Right	2.73	0.018	(1.06, 19.37)
Top	4.14	0.004	(8.74, 34.05)
Front	4.85	0.004	(9.09, 27.66)

Table 5.1: Control group orientation task: t, P-value and range of time improvement with 95% confidence. Range time in seconds.

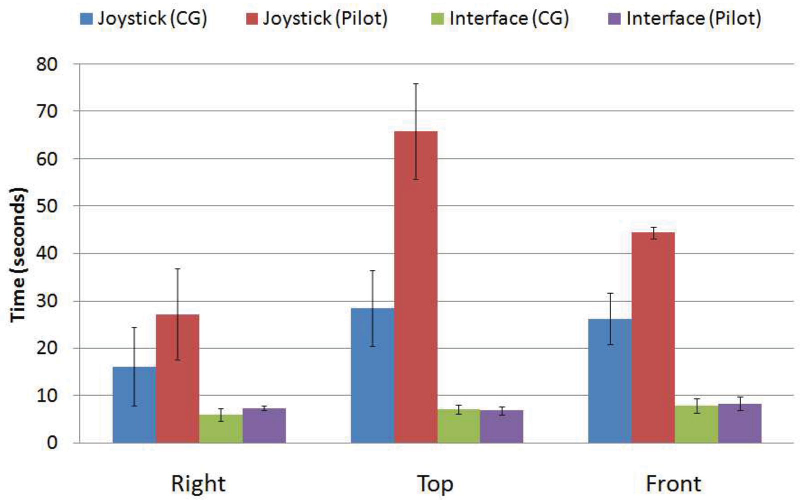


Figure 5.10: Average time to complete orientation. When performing an orientation task our pilot study user was faster by 73%, 89% and 81% when using our interface to orient from the right, top and front respectively. When performing an orientation task the control group users were faster by 63%, 75% and 70% when using our interface to orient from the right, top and front respectively.

some cases by a factor of 5 (compare in Figure 5.10, pilot user visual interface time with pilot user joystick time). Similarly the gap between the pilot user and the control group is also reduced.

5.3.3.2 Drinking task

A cup was surrounded by different obstructing objects (see Figure 5.11). The user was asked to bring the cup to a position where she can drink from it. The user was also told that the cup was full of water and that her objective was to not spill the water during the process of bringing the cup close to her mouth. The user was free to decide what obstructions to remove and grasp to use. Two control methods were used. Each user performed the task three times per control method. To avoid bias, order of selection of the control method was randomized for each user. The two modes

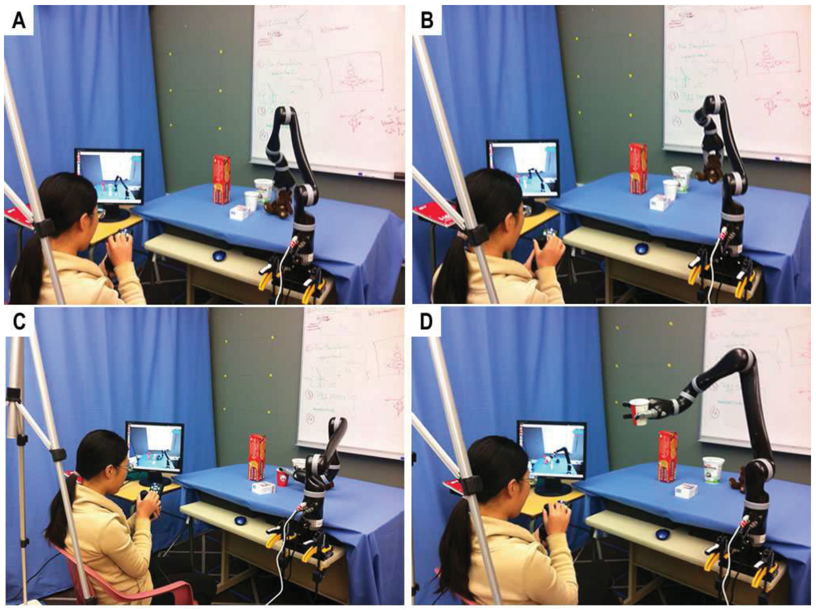


Figure 5.11: Drinking Task.
A cup is surrounded by obstructing objects, the user was asked to bring the cup to a position where she can drink from it. In this particular trial the user decides to moved first the teddy bear to unblock the cup (A) and (B). Then, orient the hand in a right grasping position, pick up the cup and finally bring it close to her (C) and (D).

used were: (1) Control the arm with direct teleoperation through JACO's built in Cartesian controller driven by a joystick. (2) Alternate the arms control between direct teleoperation and an assisted control scheme where our visual interface was used to position the arm, and the joystick controller was used to finalize the grasping task. A comparison between the direct teleoperation using the joystick and the mixed joystick-interface approach is shown in Figure 5.12. The pilot user performed slightly better with joystick-interface than by only using the joystick. However, the control group performed better using only the joystick. A possible explanation of the results is that shifting between two devices may increase execution time. The better performance of the pilot user is explained by his experience in using his regular power wheelchair joystick and constantly shifting to his mouse pad for internet browsing.

5.3.3.3 Opening a Jar and opening/closing a drawer

We demonstrate the effectiveness of the pull/push and rotating modules of our interface with two example applications: "opening a jar" and "opening/closing a drawer"

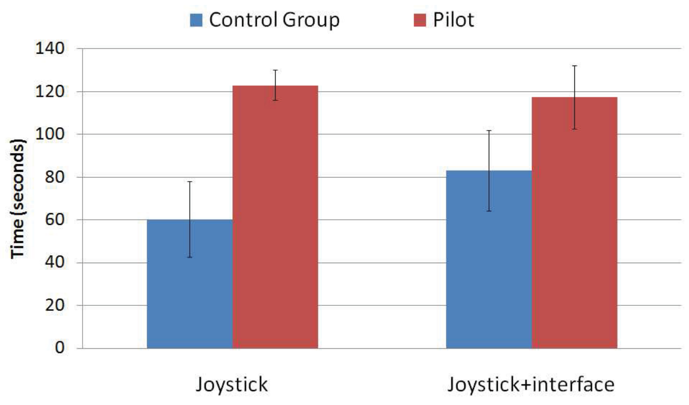


Figure 5.12: Time to complete the drinking task for the joystick and a mixed mode using the joystick and our proposed vision interface.

(See Figure 5.5). In the first experiment the user was asked to remove a lid from a plastic jar using both joystick command control and our proposed compliant task selection interface 7 times. The average completion time using the interface was 127±20 seconds while using the joystick the average time was 223±14 seconds. In the second experiment the user was asked to open a drawer and then close it 7 times. The average completion time using the interface was 76±8 seconds while using the joystick on average was 118±9 seconds. The complaint nature of controllers helps the user to complete the task successfully in the presence of inaccurate position/orientation of the end-effector. Demonstration of our interface for these tasks can be seen in the supplementary videos [106].

5.3.3.4 Subjective analysis

At the end of the experiment, users were asked to fill a questionnaire. In the first section, the user rates in a Likert-type scale from 1 to 7 the difficulty of completing both tasks with the two different interfaces. The results are shown in Figure 5.13. Both tasks where completed by all users during our trials. During the *orientation task*, both pilot user and control group perceived the joystick at least 2 times more difficult to use than the interface. In the *drinking task*, the joystick-interface was perceived easier than the only joystick control mode. In general, users perceive the use of the interface easier than the direct joystick teleoperation. Something interesting to notice is that, although the time performance in the control group was better using only the joystick in the drinking task, the user perceived less difficulty when the interface was used. Thus indicating that faster performance does not necessarily reflect a better system.

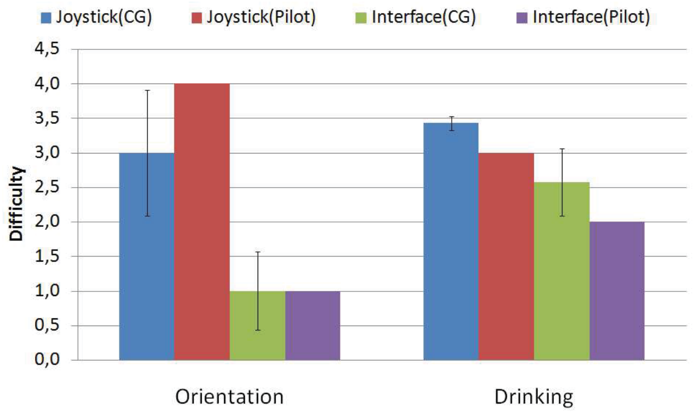


Figure 5.13: Users rate on a scale from 1-7, the difficulty of completing the orientation task and the drinking task using the joystick and our vision interface

The last question consists in rating how physically and mentally demanding the use of both interfaces was. The result is shown in Figure 5.14. It is clear that the use of our vision system is less demanding physically and mentally for both the pilot user and the control group. This is also expected because, as we mentioned, through our 2D interface, we hide the complexity of controlling a robot-arm.

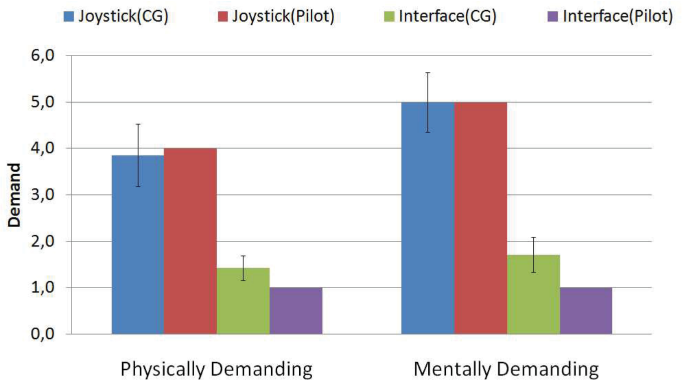


Figure 5.14: Users rated on a scale from 1-7 the physical and mental demand on completing the orientation task and the drinking task

5.4 Interface Improvements and Discussion

The user study and informal comments from the participants lead to several improvement suggestions for our system. The overall feedback with respect to the interface

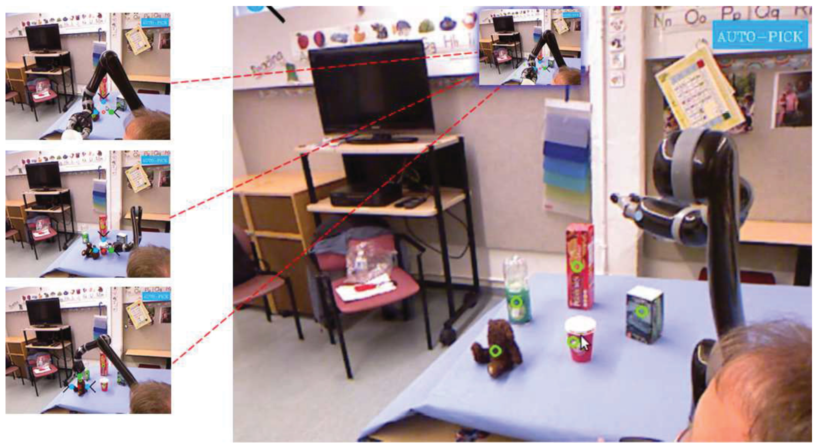


Figure 5.15: Interface preset mode. Left: three preset locations have been saved: (1) Near the user face to take a sip; (2) Picked location for the cup; (3) On top of the teddy bear. Right: The user can circle through the images of the preset list and display them as an inset on the main video window, then select the pre-set location he or she desires.

from the pilot users and control group was that the simplicity of the interface makes the interaction intuitive. However, some users suggested that the interface can offer more functionalities without loosing simplicity.

5.4.1 Preset positions

In particular our second pilot user comment that: "It might be good to have other preset positions besides "home". If a drink is picked up, perhaps a preset near the face to take a sip, then remember where it was picked up from to return it to exactly where it was, like a "last position" button. Then the drink could be chosen easily any time during a conversation for example". Current wheelchair mounted arms allow positions to be recorded and recalled. However, the interface is button-based and cumbersome to the user, who in addition to learning several modes to teleoperate the robot arm, has to learn a new combination of buttons to access stored positions. A natural way for humans to select a stored robot position is through image icons. Figure 5.15 shows our proposed interface. When a robot pose is stored, the camera also stores an image of that position in the current environment. To recall a pose, the user enters recall mode and scrolls through the image icons until the desired one is found. The icons are shown as a small inset in the main video screen.

With the Kinect sensor it is easy to obtain calibrated Euclidean measurements registered with the image. This allows the user to click on any two image points and have the system report the metric distance between the corresponding physical points. We tested the accuracy of by placing rulers in the image (Fig. 5.16), and found it to have cm accuracy. A practical use would be for a handicapped to e.g. check that he can pass through a doorway 5.16, or that the hand size and reach is suitable to grasp an object in a cupboard.

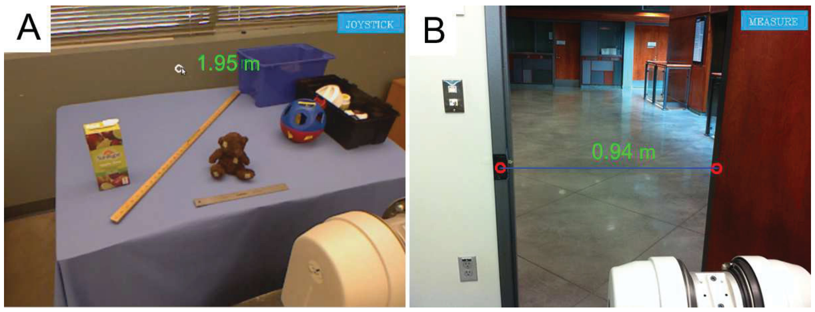


Figure 5.16: A) By clicking in any point in the image the distance to that particular location is found, this is useful to the end-user to know if a particular object is inside the robot's workspace. B) By clicking in two image points the 3D distance between them is display; this feature is desirable to anticipate physical restrictions in the space, e.g., a door with a width less than the require to pass with the power wheelchair.

5.4.2 Small Object Manipulation Tasks

Maheu *et al.* [22] evaluated the JACO robotic arm in teleoperation mode with 31 users with upper extremity disability and concluded that using it in a daily base could reduce care-giving time by 41%. Our system would improve both cognitive load and task completion time for this percentage of tasks. Our system may also make more tasks efficiently doable for disabled users than what is practical with the current joystick tele-manipulation. However this study only takes medium size objects into account. Many tasks of daily living require small object manipulation and more dexterity. Our system can be used to assist grasping in horizontal, tilted and vertical planes, e.g, grasping from a table, floor or shelf. Currently our system automates the 6DOF translation and rotation needed to reach and align for a particular object grasping. After this the finalization of the grasping is performed by the end-user. In the majority of the cases this final stage consists of approaching the object by keeping the same orientation and then closing the hand. This works well in completing mid-size object grasping , e.g., grasping a cup, cereal box, teddy bear toy, etc. However, small objects can be difficult to detect. Figure 5.17 illustrates this problem. The depth sensor resolution is not fine enough to capture the drinking straw.

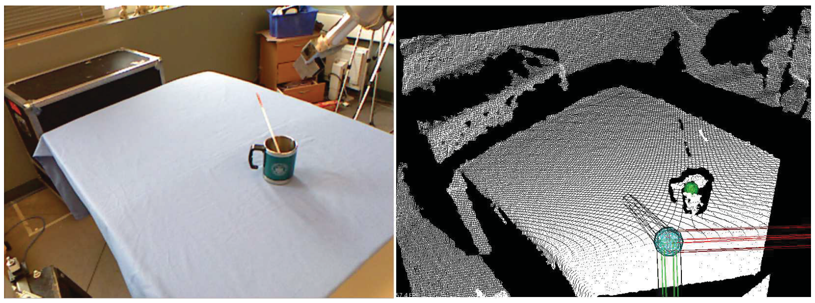


Figure 5.17: Both RGB and point cloud visualization of a scene with a mug and a straw are shown. Notice that in the point cloud visualization the straw is barely observable.

In the 1D or 2D list selection modes and in the general pick mode the user can approximate by clicking near the object, but the alignment would not be center with the object. Figure 5.18 shows an example where the user intention is to grasp the straw. To get close to the target object, the user clicks on the mug and select the right grasping approach (Figure 5.18A). Although the location is not to far from the grasping object the orientation is not the best and the user may struggle to finish the grasp. An alternative is shown in Figure 5.18B. An eye-in-hand camera is integrated in the system. After getting close to the mug, the system switches to using the eye-in-hand images and the user clicks on the straw (red ring Figure 5.18D). The user can then finish the grasp with semi-autonomous assistance guiding the motion direction (bearing-only visual servoing) towards the straw (Figure Figure 5.18E,F).

Tsui *et al.* [7] allowed visual servoing but was restricted to a-priori defined objects, where a grasp for each of the defined objects had been pre-recorded. The system does not work for arbitrary objects, only pre-trained objects. this can be an issue since humans live in unstructured environments, and new objects appear all the time. Note “new” here would be an object that can have only minor differences. If Tsui’s system has been trained on one brand of soda, it cannot generalize this to a can of soda with identical shape, but a different label, e.g. a different brand. The system we described and evaluated in sections 5.2 and 5.3 functions for arbitrary objects and no pre-training with an a-priori object data-base is needed. The experiments in Section 5.2 uses vision to assist with the reach, but gives the user control of the grasp. It can however be modified to allow vision assisted grasping and fine manipulation by adding an camera mounted in the hand (Figure 5.18B). Once the arm has reached to near the object, this camera gives a detailed view of the object and grasping situation. Here the user can visually select a second motion specification defined in the detailed view. This motion can be carried out either using visual servoing or calibrated stereo vision. Hence the difference between Tsui's and our system is that Tsui can handle a set of pre-trained objects with a one step specification, while our proposal uses a two step, coarse, then close up specification to handle arbitrary objects in unstructured environments. An approach for doing this consists on image task specification [72] where the user specified the robot movements by using geometric primitives, the challenge here is to find a good interface to do so. In practice we have found this system to be intuitive [107]. The main challenge in this area is adapting the system such that we can minimize the amount of constraints the user must specify before a task can be performed.

5.5 Conclusions

We designed and developed a computer vision system aimed to allow upper body disabled people to use a robot arm. Our system was implemented on two different robots: Barrett's WAM arm and Kinova's JACO arm. The precision of our system is 1.78 ± 0.67 cm when reaching objects in the workspace plane. Our system was refined through discussions with disability robotics company Kinova and one disabled robot arm end-user.

Experiments were performed with a pilot study user and 8 participants in a control group. Our vision system on average was faster than the direct joystick control in achieving orientation tasks. The vision system on average was 1.81 and 1.69 times faster than the direct joystick control in achieving orientation tasks for our pilot study and our control group respectively. While performing a drinking task, it was slightly faster in the pilot study and slower in the control group. Participants rated the vision system to be easier to use than direct joystick control of the arm for all cases. Pulling/pushing and rotating compliant capabilities of the interface were tested experimentally by opening/closing a drawer and removing a lid from a jar. Our obtained results suggest that our system would be helpful to disabled users of wheelchair mounted robot arms such as Kinova's Jaco. We present future directions to increase the functionality of our system without removing its simplicity.

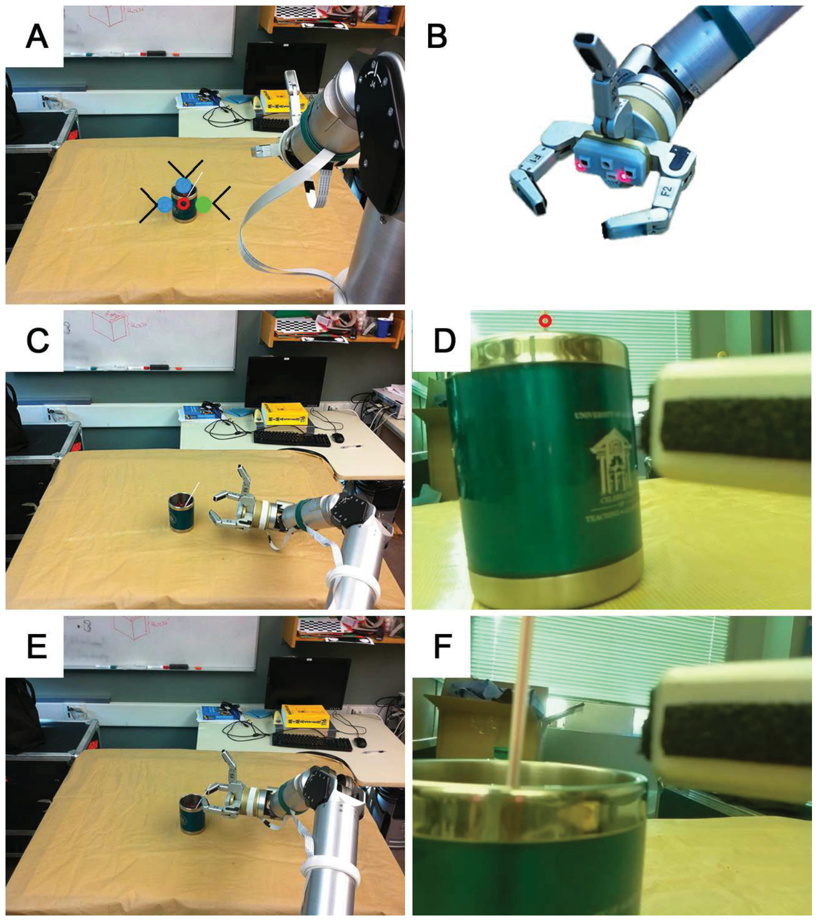


Figure 5.18: A) The user clicks on the mug and choose the right grasp approach. B) Two eye-in-hand cameras were added to our 7DOF WAM robot. C) The robot moves to the grasping location. D) From the eye-in-hand camera view the user click on the target object. E) The user can reposition the robot based on the pixel clicked. F) Eye-in-hand view after reposition

Chapter 6

Remote Display-based pointing in Tele-Manipulation

The general task that we are addressing is to tele-operate a robot manipulator while guiding it through a desired path (see Fig. 6.1). In these applications, direct bilateral teleoperation is often preferred as it results in better operator's performance when compared to unilateral control. Our hypothesis, however, is that by introducing a vision-assisted control system which is capable of generating VFs, the operator's performance in the unilateral case can be improved, perhaps up to a point where performance is similar to the bilateral configuration. We present the design and implementation of a flexible force-vision-based interface, allowing local operators to visually specify a path constraint to a remote robot manipulator in an on-line fashion during the teleoperation. Using bilateral and unilateral configurations, we compare our system to direct teleoperation through user studies. Three performance metrics (smoothness, error and execution time) and a subjective evaluation (NASA TLX) were used to quantify user performance. The trials show that our system outperforms direct teleoperation and reduces cognitive load. Our findings show that the performance of a unilateral teleop configuration with visual-force constraints surpass a bilateral teleoperation configuration in terms of displacement error and variance, as well as allowing users to complete tasks faster and with a smoother trajectory.

6.1 Introduction

Tele-robotics has brought humans the capacity to change the physical world remotely. This has made a big impact on many fields. Some examples are: Surgery, disaster response robotics, under-water and space exploration. Motivated by the Fukushima incident, DARPA conducted the robotics challenge where robots controlled through semi-autonomous teleoperation completed tasks hazardous to humans. Yanco *et al.* [108] study the different group performances during the challenge. They concluded that the use of effective semi-autonomous routines, in combination with sensor fusion, and easy to use interfaces for users lead to better performance. Perhaps the most desirable capabilities to combine during robot manipulation are vision and force. Common vision-force control approaches use impedance-based and hybrid-based strategies [109]. In the former a vision based controller drives the movement of the robot's end-effector and the force controller provides compliance to external disturbances [110]. While, in the latter, vision and force controllers are working in parallel and thus it is necessary to ensure orthogonality between both controllers to avoid any conflict at the actuator level [111, 112]. The main role of vision-force control in tele-manipulation is to assist the operator by generating motion guidance active constraints also known as *Virtual Fixtures* (VF).

These constraints are typically used to either restrict the movement of the robot to certain regions (forbidden region VF) or to guide the operator through a specific path (guidance VF) [113, 114]. VFs reduce the task complexity and operator's workload. They have been used in many applications, for example polishing/finishing mechanical parts [115], automotive inspection [116], robot-assistive surgery [117, 118, 119], human-robot cooperative tasks [120] or space exploration [121, 122, 123].

One of the biggest challenges in applying VFs in real world tele-robotics scenarios is to find a way to provide a flexible and efficient geometric constraint specification [113].

For example, Jiang *et al.* [124] propose a flexible VF that can be adjusted in the presence of an obstacle in the path. In [120], an autonomous error compensation method is presented to overcome human difficulties in simultaneously controlling the position and orientation of a 6-DOF robot under reference direction fixtures [120]. Authors of [120] proposed a dynamic VF for robot-assisted surgery. While there are several attempts, geometric constrain specification is still a challenging problem [113]. We address this problem through the design of a novel 2D interface that allows the

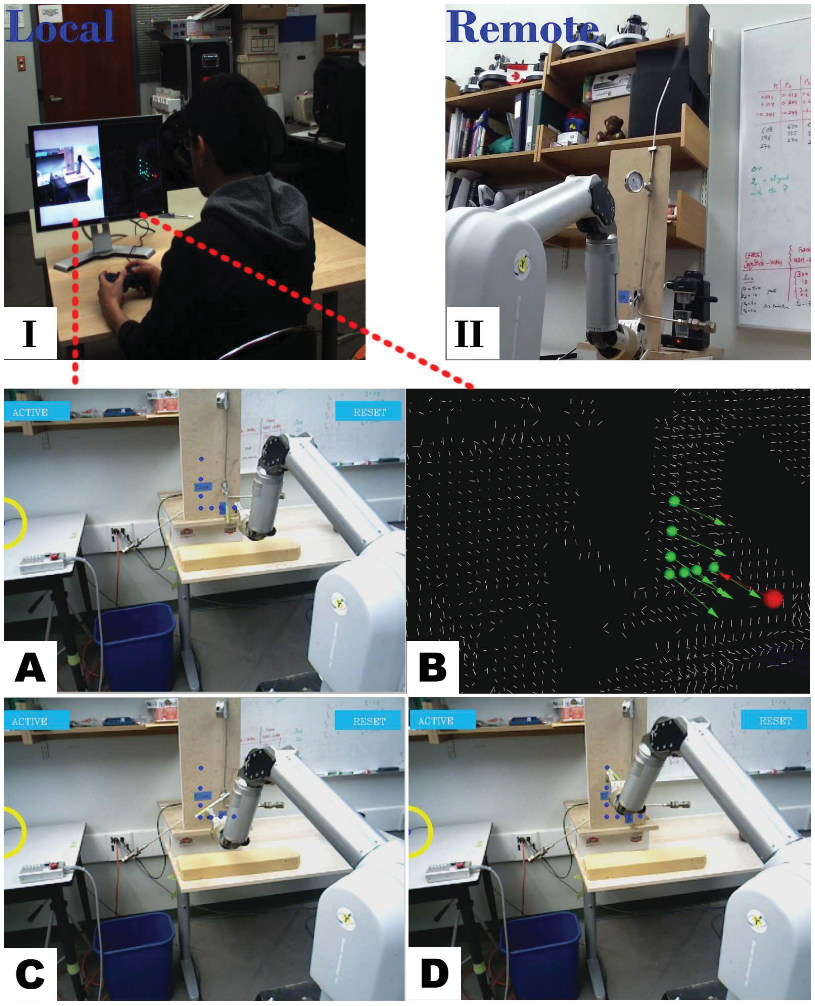


Figure 6.1: The use of our proposed interface is illustrated by closing a steam valve in a remote location. I) The local operator is provided with a monitor displaying our interface and a game-pad for interaction. II) In the remote site a 7-DOF WAM arm is teleoperated with the aid of our interface to close a steam valve. **A** and **B** show the RGB and Point Cloud visualization of our interface. A) The operator defines a desired path in the 2D image by clicking (blue dots). B) The 3D path and the associated normals to the surface are calculated and visualized (green spheres and green arrows). C, D) Once the constraints are activated the user can complete the task by simply controlling 2-DOF: motion through the path direction and motion normal to the surface.

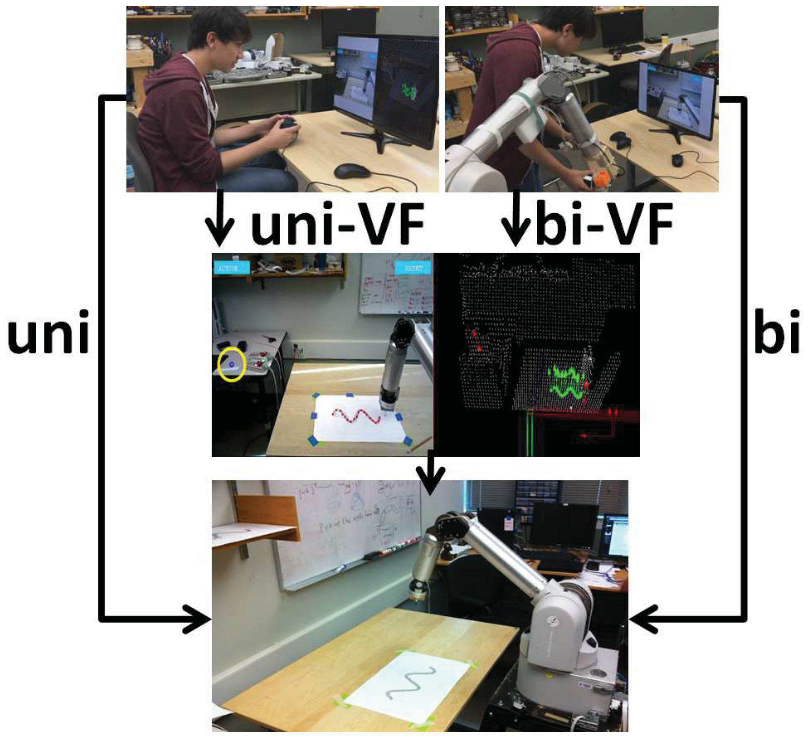


Figure 6.2: Different teleoperation configurations. 1) uni:gamepad to WAM 7-DOF; in this configuration there is not force feedback. 2) bi: 4-WAM with 3-DOF gimbal to WAM 7-DOF; in this configuration there is force feedback. 3) uni-VF: uni configuration with aid of the interface. 4) **bi-VF:bi configuration with aid of the interface.**

operator to intuitively generate a 3D VF for the remote manipulator on top of the image stream coming from the remote location. Furthermore, we present the design of a tele-manipulation system that aims to improve an operator's performance and reduce his/her workload independently of the master device.
The rest of this chapter is organized as follows: section 6.2 outlines the problem formulation and the overall system description and development. Section 6.3 describes the experimental setup, procedure, results and discussion on findings. Conclusions and future works are presented in section 6.4.

6.2 System Description

In order to test our hypothesis, we have implemented different teleoperation configurations, see Fig. 6.2 :

- Direct Unilateral (*uni*): This is a unilateral configuration where the local operator controls the remote robot through a game-pad. A Cartesian force controller is actuated through a game-pad to control the remote robot.

- Direct Bilateral (*bi*): Both local and remote robots are linked. This configuration allows the local operator to control each joint of the remote robot by physically manipulating the corresponding joint of the local one. We use a similar approach to the one proposed by Glover *et al.* [125] to provide haptic feedback to the local operator from the remote robot.

- Unilateral with Virtual Fixture (*uni-VF*): Similar to unilateral configuration plus VF constraining the remote robot's end-effector to the specified path.

- Bilateral with virtual fixture (*bi-VF*): The same linking described in the Bilateral configuration. In addition, the remote robot is constrained to a path specified using our force-vision module.

In summary our system implementation provides both unilateral and bilateral teleoperation for a 7-DOF WAM arm. In the unilateral case the remote robot arm is tele-operated with a game-pad, while in the bilateral case a 4-DOF WAM arm with a 3-DOF gimbal is used as the master manipulator. To provide the path following functionality to the 7-DOF WAM arm, our system relies on three modules: *Visual Interface*, *Control* and *Vision* as shown in Fig. 6.3.

6.2.1 Visual Interface Module

The *Visual Interface* (Fig. 6.4) is equipped with a drawing tool that allows the operator to define a desired trajectory on the image stream from the remote site. The interface also provides a real time point cloud visualization in which the operator can corroborate visually the selected 3D path and the corresponding estimate of the surface normals.

6.2.2 Vision Module

The *Vision module* streams the video coming from an RGB-D sensor in the remote location to the visual interface presented to the operator. In our tests we utilize a Kinect sensor. This module feeds the *Control module* with the 3D path coordinates and corresponding surface normals with respect to the robot reference frame. The

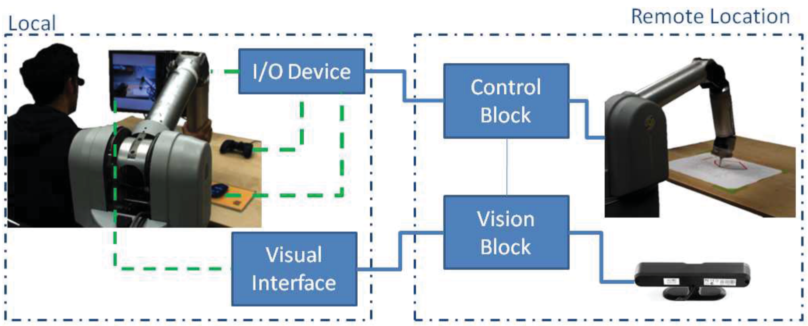


Figure 6.3: System overview. Left: The I/O devices that the local operator interacts with, are the mouse, screen, game-pad and 4-DOF WAM arm. The first two are used for monitoring the remote location and for specifying the desired path constraints. Game-pad is used for the unilateral teleoperation and the 4-DOF WAM arm for the bilateral teleoperation. Right: The vision and control module are in-charge of processing the RGB-D sensor information and constraining the 7-DOF WAM arm to the specified path.

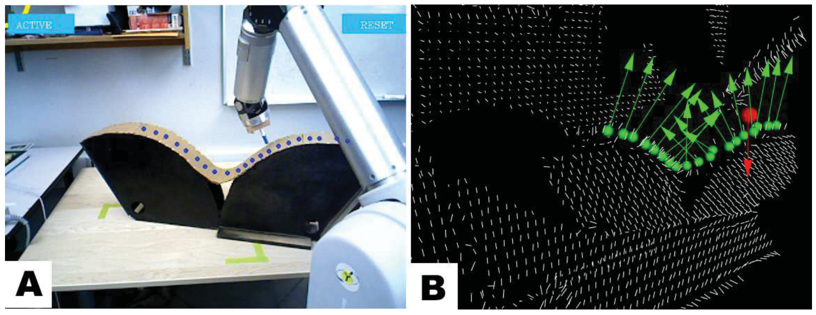


Figure 6.4: Visual Interface: A) By clicking on the image the user set a path on top of the surface. B) The scene point cloud is visualized with the selected path (green spheres) and the calculated normals (green arrows).

3D path coordinates are found using the direct correspondence of the RGB sensor and depth sensor. These are then converted into robot coordinates through a Kinect-Robot transformation matrix (previously calculated using a calibration routine). The typical approach to calculating surface normals consists of fitting a plane to the neighborhood P_i of the target point p_i. P_i is either formed by the k nearest neighbors of p_i or by points within a radius r from p_i. Given P_i, its covariance matrix $\boldsymbol{C}_i \in \mathbb{R}^{3x3}$ is computed (see [82] for details of the computation). The eigenvectors of the covariance matrix, $\boldsymbol{v}_{i,0}$, corresponding to the smallest eigenvalue $\lambda_{i,0}$, can be used as an estimate of the target normal $\hat{\boldsymbol{n}}_i$ [82]. To reduce the computational time and achieve real time performance, we used the integral normal estimation approach.

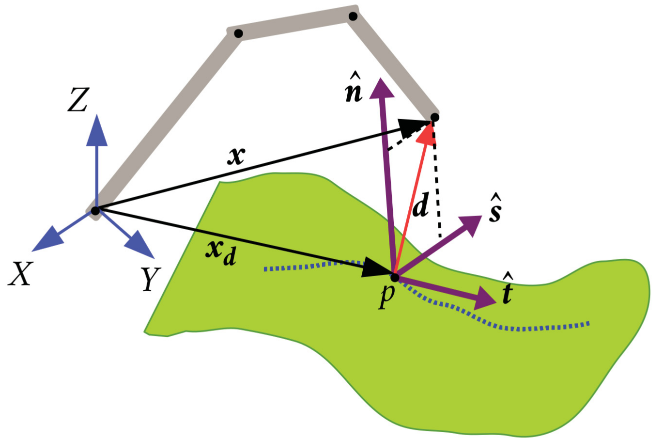


Figure 6.5: Schematic of the path on a surface and construction of the corresponding path reference frame.

This implementation takes advantage of the organized structure of the point cloud acquired by the RGB-D sensor and also uses a pixel neighborhood instead of a spatial neighborhood. By using the pixel neighborhood the simplest approach is to use the right-left pixels and up-down pixels to form two local tangential vectors and calculate the normal using the cross product. However, since the data is noisy due to the nature of the sensor, an average tangential vector calculation (known as a integral image [101]) is performed. In our implementation we use the PCL [86] *Average 3D Gradient* mode which creates 6 integral images to compute smoothed versions of horizontal and vertical 3D gradients and computes the normals using the cross-product between these two gradients [102].

6.2.3 Control Module

This section describes the details of the controller module for remote robot. Our robot is a torque controlled 7-DOF WAM manipulator with the dynamic equation of

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} + \boldsymbol{\tau}_{ext}$$

where $\boldsymbol{q}$ is the joint angles, $\boldsymbol{M}(\boldsymbol{q})$ is the positive-definite inertia matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is the Coriolis matrix, $\boldsymbol{g}(\boldsymbol{q})$ is the gravitational force, $\boldsymbol{\tau}$ the actuators torques, and $\boldsymbol{\tau}_{ext}$ is the external generalized force applied to the robot by the environment.

As shown in Fig. 6.5, the goal of the controller is to constrain the robot end-effector to the path (on a surface) while maintaining a contact force (F_n) on the

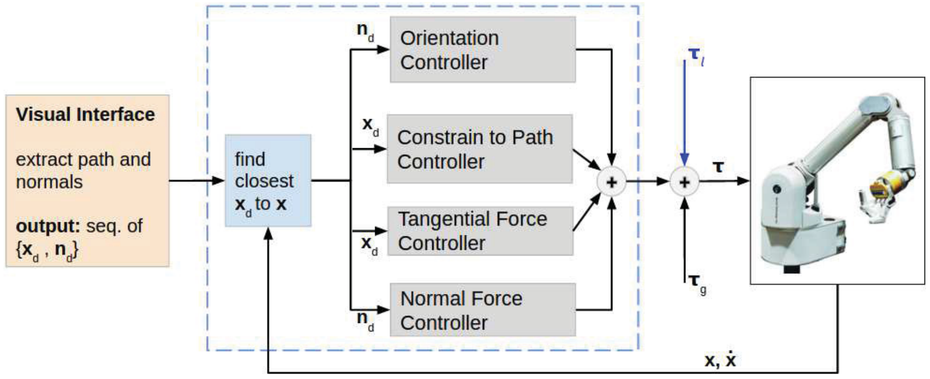


Figure 6.6: Control architecture of the remote robot.

surface. This is achieved by projecting the motion onto orthogonal directions along the tangential ($\hat{\boldsymbol{t}}$), normal ($\hat{\boldsymbol{n}}$) and side directions ($\hat{\boldsymbol{s}}$), see Fig. 6.5. A dedicated controller is then designed for each direction. An additional controller is also required for orienting the robot's tool. The proposed control architecture is shown in Fig. 6.6 and it is composed of a *Constraint Evaluation* module and four sub-controllers. It is assumed hereafter that the visual interface module provides the desired path, i.e. a collection of 3D points on the surface ($\{\boldsymbol{x}_{d_i}\}$). Then, the inputs to the controller are (i) the desired path ($\{\boldsymbol{x}_{d_i}\}$) and (ii) the corresponding normal directions at each point ($\{\hat{\boldsymbol{n}}_i\}$). Details of each of the modules are given next.

6.2.3.1 Constraint Evaluation

The initial step to decide on the direction of anisotropy of the active constraint, is to find the relative configuration of the robot and the constraint [113]. Following the methodologies of [114], the desired point $\boldsymbol{x}_{d_i}$ is the closest point on the reference path to the robot's tool tip $\boldsymbol{x}$. Thus, for any given $\boldsymbol{x}$, the corresponding $\boldsymbol{x}_d$ is obtained from:

$$\boldsymbol{x}_d(\boldsymbol{x}) := \arg \min_{\hat{\boldsymbol{x}} \in \{\boldsymbol{x}_{d_i}\}} \|\hat{\boldsymbol{x}} - \boldsymbol{x}\| \tag{6.1}$$

6.2.3.2 Path controller

Given the desired path, the tangential direction at each point is computed first by $\hat{\boldsymbol{t}}_i = \boldsymbol{x}_{d_i} - \boldsymbol{x}_{d_{i-1}}$. From this and the given normal direction, the path reference frame is fully characterized by $\{\hat{\boldsymbol{n}}, \hat{\boldsymbol{s}}, \hat{\boldsymbol{t}}\}$, where $\hat{\boldsymbol{s}} = \hat{\boldsymbol{n}} \times \hat{\boldsymbol{t}}$ (see Fig. 6.5). Path controller is responsible for forcing the end-effector to be constrained to the path in the $\hat{\boldsymbol{s}}$ direction.

To this end, let $\boldsymbol{d} = \boldsymbol{x} - \boldsymbol{x}_d$ be the minimum distance of the end-effector tip to the path from (6.1). An impedance controller is then used to minimize the projection of $\boldsymbol{d}$ onto $\hat{\boldsymbol{s}}$ direction. To this end, define

$$F_s = K_{p_p}\big((\boldsymbol{x} - \boldsymbol{x}_d) \cdot \hat{\boldsymbol{s}}\big) + K_{d_p}(\dot{\boldsymbol{x}} \cdot \hat{\boldsymbol{s}}) \tag{6.2}$$

as the magnitude of the side direction force, due to the virtual spring K_{p_p} and damper K_{d_p} system that pulls $\boldsymbol{x}$ towards point $\boldsymbol{p}$ on the reference path. The torque command that generates this force is

$$\boldsymbol{\tau}_p = \boldsymbol{J}^T(F_s\,\hat{\boldsymbol{s}}) \tag{6.3}$$

where $\boldsymbol{J}$ is the task Jacobian matrix of robot.

6.2.3.3 Normal Force controller

This compliance controller is responsible for maintaining the contact with the surface by applying a normal force (F_n) in the $-\hat{\boldsymbol{n}}$ direction:

$$F_n = K_{p_n}\big((\boldsymbol{x} - \boldsymbol{x}_d) \cdot \hat{\boldsymbol{n}}\big) + K_{d_n}(\dot{\boldsymbol{x}} \cdot \hat{\boldsymbol{n}}) + F_0 \tag{6.4}$$

$$\boldsymbol{\tau}_n = -\boldsymbol{J}^T(F_n\,\hat{\boldsymbol{n}}) \tag{6.5}$$

where F_0 is the nominal desired force to be applied on the surface.

6.2.3.4 Tangential Velocity controller

This controller drives the robot's end-effector in the path direction with a desired tangential velocity v_d:

$$F_t = K_{p_t}(\dot{\boldsymbol{x}} \cdot \hat{\boldsymbol{t}} - v_d) + K_{d_t}(\ddot{\boldsymbol{x}} \cdot \hat{\boldsymbol{t}} - \dot{v}_d) \tag{6.6}$$

$$\boldsymbol{\tau}_t = \boldsymbol{J}^T(F_t\,\hat{\boldsymbol{t}}). \tag{6.7}$$

6.2.3.5 Orientation controller

Lastly, an additional controller is needed to align the end-effector to the desired orientation. To this end, the following compliance controller is used

$$\boldsymbol{\tau}_o = \boldsymbol{K}_{p_o}(\boldsymbol{o} - \boldsymbol{o}_d) + \boldsymbol{K}_{d_o}(\dot{\boldsymbol{o}}) \tag{6.8}$$

where $\boldsymbol{o}_d, \boldsymbol{o}$ are the desired and actual orientation of the robot's end-effector. In the general case, the desired orientation $\boldsymbol{o}_d$ is path dependant. Here we assume that the end-effector (tool) reference frame $\{X_t, Y_t, Z_t\}$ is reoriented such that Z_t is aligned with -$\hat{\boldsymbol{n}}$ and Y_t is aligned with the tangential direction $\hat{\boldsymbol{t}}$. The rotation matrix from tool to base is then $\boldsymbol{R}_t^b = [\hat{\boldsymbol{s}}, \hat{\boldsymbol{t}}, \text{-}\hat{\boldsymbol{n}}]$ from which $\boldsymbol{o}_d$ is obtained.

The final actuator torque command $\boldsymbol{\tau}$ that is sent to the robot is

$$\boldsymbol{\tau} = \boldsymbol{\tau}_g + (\boldsymbol{\tau}_c + \boldsymbol{\tau}_n + \boldsymbol{\tau}_t + \boldsymbol{\tau}_o) + \boldsymbol{\tau}_l \tag{6.9}$$

where $\boldsymbol{\tau}_g = \boldsymbol{g}(\boldsymbol{q})$ is the gravity compensation torque and $\boldsymbol{\tau}_l$ is the torque command received from the local end (either from the game-pad in unilateral configuration or from the WAM in bilateral configuration). In the bilateral configuration,
$\boldsymbol{\tau}_l = \boldsymbol{K}_{p_l}(\boldsymbol{q}_r - \boldsymbol{q}_l) + \boldsymbol{K}_{d_l}(\dot{\boldsymbol{q}}_r - \dot{\boldsymbol{q}}_l)$.

6.3 Experiments

The experiments consist of two main tasks: drawing on a surface and a circular alignment. Users are required to maintain the contact with the surface during the tasks. We conducted a pilot study with 9 users (2 female and 7 male). The tests were approved by University of Alberta Research Ethics and Managements (Pro00054665). All of the users have normal or corrected to normal vision. The complete experiment took about 90 minutes per participant. For each task users utilize all 4 different system configurations described in section 6.2. The sequence of the tasks and the teleoperation configuration was randomized across participants. At the end of the experiment the users were asked to fill a 21 point NASA TLX form.

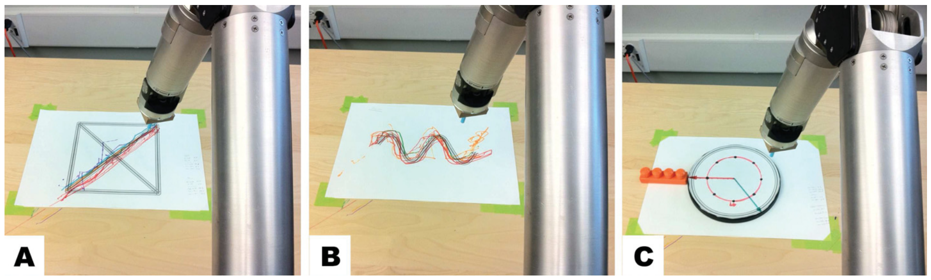


Figure 6.7: Three different experiments were conducted using both direct teleoperation and our VF controller. From left to right: line drawing, path drawing and circular alignment tasks.

6.3.1 Controller Parameters

The following controller gains are chosen for all the interface experiments. A nominal force of $F_0 = 4\,N$ is applied for maintaining the contact.

	Stiffness Gain	Damping Gain
Path Controller	1000	10
Normal Force Controller	10	0.1
Tangential Force Controller	15	0.01
Orientation Controller	diag(7,7,0)	0.01*diag(1,1,1)
Bilateral Controller	diag(900, 2500, 600, 500, 50, 50,8)	diag(10,20,5,2, 0.5,0.5,0.05)

Table 6.1: Controller gains

6.3.2 Tasks Description

6.3.2.1 Drawing Task

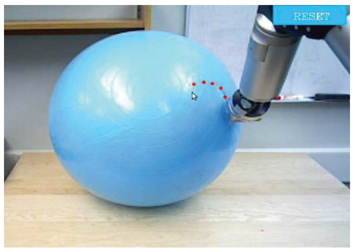
The participant is presented with two drawing patterns: a *Line* (Fig. 6.7A) and a sinusoidal *Path* (Fig. 6.7B). The drawing task consists of tele-operating the remote robot with a pen attached to its end-effector. Users have to make contact with the surface on one end of the pattern and follow it through without lifting the pen from the surface. To perform these tasks users first click on the image to define the desired path. Once defined, users manually activate the VFs to constrain the robot to the path they defined.

6.3.2.2 Circular Alignment Task

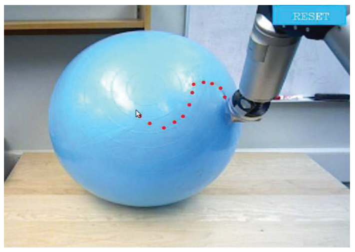
A pattern with two arrows (green and red) is placed on top of a turntable as shown in Fig. 6.7C. Initially, the red arrow is aligned with an orange marker. The task is then to align the green arrow with the marked location by rotating the turntable. To rotate the turntable, users have to make contact with the turntable first, and then maintain the contact while rotating in a counter-clockwise circular motion.

6.3.2.3 3D-writing on a Ball

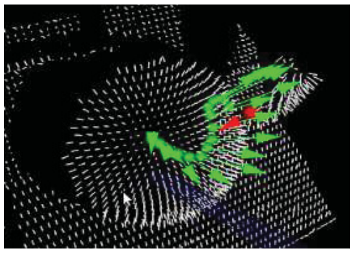
We demonstrate the capability of our proposed control architecture on a challenging task of following an arbitrary 3D path on a ball. Snapshots of the task are shown in Fig. 6.8. Demonstrations of these tasks can be seen in the attached video.



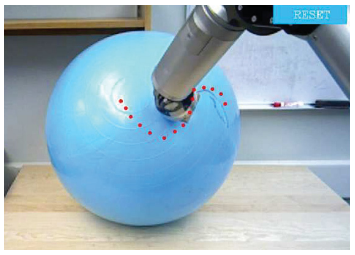
(a)



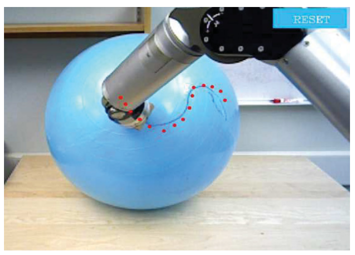
(b)



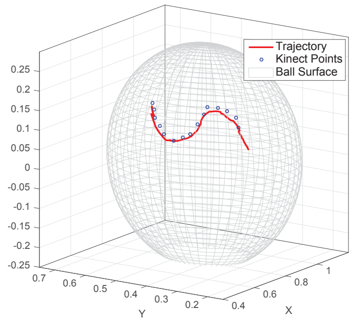
(c)



(d)



(e)



(f)

Figure 6.8: Snapshots of tele-operation during the 3D task: [a],[b] user defines the 3D path by clicking in the 2D image interface, [c] Point-cloud visualization of the desired path and the associated normals. [d], [e] user tele-operates the robot using the game-pad, while the proposed VF controller maintains the contact with the 3D surface and reorients the tool. [f] illustration of the end-effector trajectory vs. the defined path.

6.3.3 Anecdotal Examples

To proof the applicability of our interface to real world scenarios we complete two anecdotal examples: 1) Closing a steam vale. 2) Gluing a T PVC pipe.

6.3.3.1 Closing a steam valve

As a proof of concept we run a test during a robot demonstration in an Industrial Mixer at the University of Alberta Computing Science building. The task consists of closing a steam valve (see Fig. 6.9) through teleoperating a 7-DOF robot located in the department's 2nd floor, with a 4-DOF WAM arm located in our robotics lab (department's 3rd floor). Through this task we show the viability of using our system in a real task. We noticed that by planing the robot trajectory beforehand the task is carefully laid out and obstacles that were hit during direct teleoperation were avoided during the task specification (see the video attached).

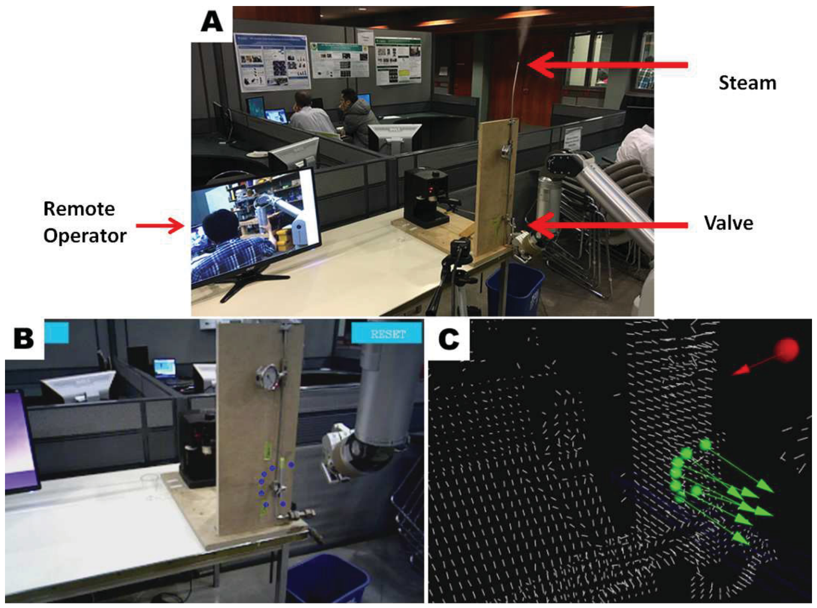


Figure 6.9: Anecdotal example during Industrial Mixer at the University of Alberta

6.3.3.2 Gluing a T PVC pipe

Our interface allows to specify different magnitudes of normal contact force to complete tasks like welding, polishing, painting, etc. which require moving through a specific path while making contact with a surface. In this example the objective is to apply glue to a crack on a PVC pipe, see Fig. 6.10.

6.3.4 Results and Analysis

While performing the task, the position and orientation of the remote robot was recorded for analysis. In addition, when using the interface the points clicked by the user (hereafter referred to as *interface path*) are recorded. Trajectories of the end-effector controlled by one of the participants are shown in Fig. 6.11 for different tasks and with different teleop configurations. The red dashed-line represents the ground truth path, which is obtained by moving the tip of the robot to the desired path and recording the pose. It is clear from Fig. 6.11 that each teleop configuration presents a different characteristic behaviour in the tasks. The following four metrics are used to characterize/evaluate the performance of teleop configurations.

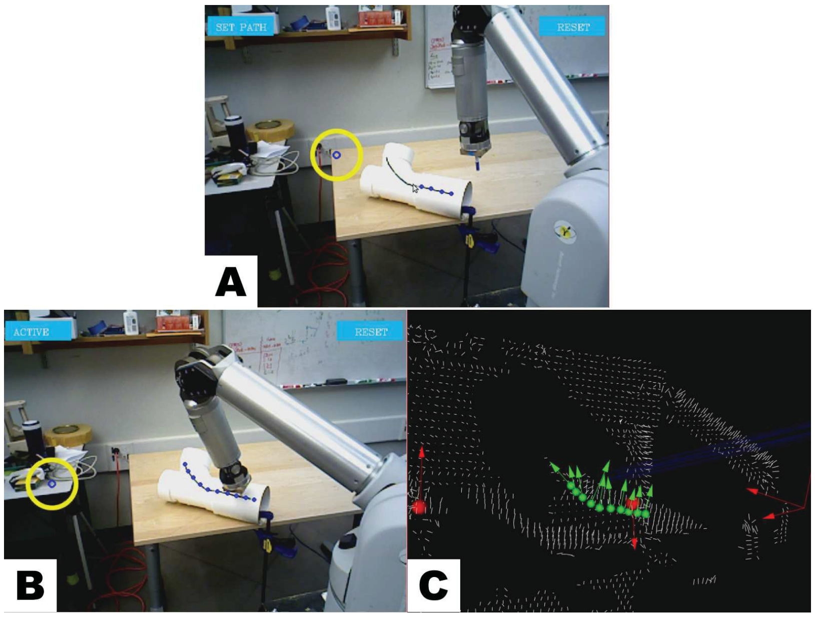


Figure 6.10: Gluing a T PVC pipe: A) The user defines a path on top of the black crack. B) Once the constraints are activated, the user can apply glue by moving the end-effector through the crack. C) The scene point cloud is visualized with the selected path (green spheres) and the calculated normals (green arrows).

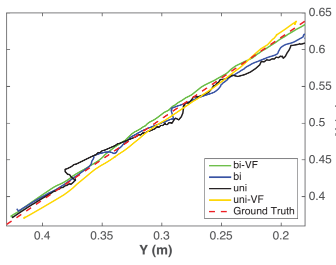
6.3.4.1 Path Smoothness

The smoothness of each path was assessed using the *Spectral Arc Length* (SAL) metric developed in [126]. SAL is a dimensionless measure of the length of the frequency spectrum curve of a speed profile over the bandwidth appropriate for the motion. For a movement with speed profile $v(t)$, SAL is calculated from

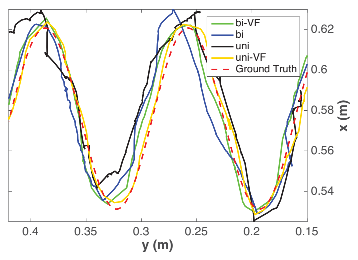
$$SAL := \int_0^{\omega_c} \sqrt{\left(\frac{1}{\omega_c}\right)^2 + \left(\frac{dV(\omega)}{d\omega}\right)^2} d\omega.$$

where $V(\omega)$ is the Fourier magnitude spectrum of $v(t)$, and $[0, \omega_c]$ is the frequency band spanned by the given movement. $\omega_c = 40\pi$ rad/s (which corresponds to 20 Hz) covers the normal and abnormal aspects of human movements [126]. It should be noted that SAL value approaches zero with increases in smoothness of the path.

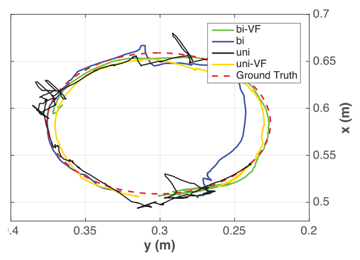
Figure 6.12 (top) compares the smoothness of the paths for different teleop configurations and for different tasks. It can be seen from this figure that VF improves the smoothness of the paths. As expected, game-pad teleop configuration results in the most jerky (non-smooth) motions. Adding the virtual fixtures significantly improves the smoothness of the paths. In fact, the smoothness of the paths with uni-VF



(a) *Line*

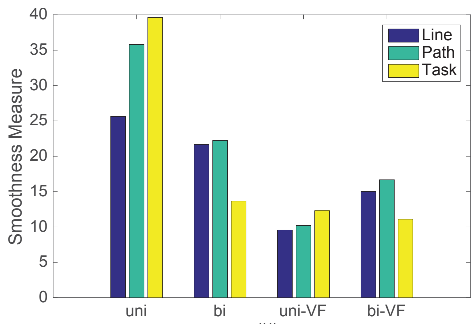


(b) *Path*



(c) *Circle Alignment task*

Figure 6.11: Sample data of a participant performing the three different tasks with the 4 different teleop configurations (uni/bi lateral teleop with/without VF). The dashed-line is the ground truth path.



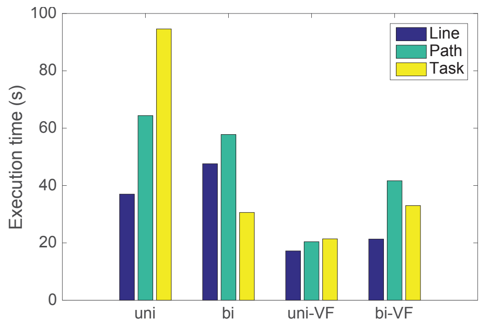


Figure 6.12: Comparison of (left) the smoothness of the paths, (right) execution time of the tasks for different teleop configurations.

configuration is (on the average) better than bilateral configuration and comparable with bilateral with VF configuration. This is due the fact that in uni-VF mode, the operator is only able to move the robot along the path.

6.3.4.2 Time to complete the task

Figure 6.12 (bottom) compares the execution time of each task for different teleop configurations. Note that the time to set the path from the interface is not considered here in the measurements as this procedure can be done offline. It can be seen from this figure that VF reduces the execution time of the tasks in unilateral configuration. Uni-VF performed the best as anticipated; adding a driving force in the tangential direction helps the operator to finish the task faster.

6.3.4.3 Trajectory Error

Table 6.2 presents the trajectory mean error (averaged over all participants) for the line and sinusoidal paths. Given the ground truth path ($\boldsymbol{P}_{GT}$), the error from the path is computed as the minimum distance of the sampled trajectory to the path, *i.e.* $e(\boldsymbol{x}) = \min_{\boldsymbol{p} \in \boldsymbol{P}_{GT}} \|\boldsymbol{x} - \boldsymbol{p}\|$. It can be seen from Table 6.2 that the bi-VF has the best performance in terms of trajectory following. While the performance of the unilateral teleop is not satisfactory, adding the VF significantly improve the performance of the uni-VF teleop up to a point that is comparable to bi-VF. Note that the error in the bilateral configuration is higher compared to uni-VF control. This observation is in line with the one of [120], because in bilateral mode, operator has to control both position and orientation simultaneously. Authors of [120] stated that this is due to the translational and rotational components of motion being decoupled from each other in such a way that the user, focusing on moving one, will not notice an error in the other and vice versa. The same trend is observed in the variance of the errors, except that the variance of the error is smaller for the uni-VF. This could be due the fact that in the uni-VF mode, the operator is only able to move along the path direction. In the bi-VF mode, however the operator is able to fight with VF and move in other directions.

		uni	bi	uni-VF	bi-VF
Line	mean error (mm)	13.78	11.76	4.50	**4.37**
	variance (mm)	8.15	5.96	**1.52**	2.19
Path	mean error (mm)	13.99	11.13	9.08	**8.51**
	variance (mm)	9.58	7.53	**4.07**	3.42

Table 6.2: Comparison of the trajectory error and variance for different teleop configurations

6.3.4.4 Loss of Contact

Table 6.3 compares the average number of times that the robot's end-effector tip lose the contact with the surface during the experiments. In the simple uni-lateral control, it is hard for the operator to maintain the contact with the environment due to the lack of haptic feedback through the game-pad. Contact control improves in the bilateral case as the user is able to feel the force feedback from the remote site and tries to maintain the contact. Table 6.3 shows the effectiveness of the normal force controller in maintaining the contact with the environment.

	uni	bi	uni-VF	bi-VF
Line	2.71	0.75	0	0
Path	3.33	1.14	0.28	0

Table 6.3: Average number of loss of contact with the surface during the line /path following tasks

In summary, the the performance of the unilateral with VF in terms of error, execution time, smoothness of the path and contact control is comparable to the one of bilateral control with VF. This also signifies that lack of haptic feedback in the unilateral control can be compensated by proper integration of vision and force control.

6.3.4.5 Operator Workload Measure

Figure 6.13 shows the mean ratings of each workload category reported by the users in the NASA TLX survey. The uni-VF configuration resulted in less overall workload than all the other configurations including the bi-VF. This again confirms our hypothesis that by introducing vision-force assistive constraints, the operator's performance in the unilateral case can be improved, perhaps up to the point where performance is similar to the bilateral configuration.

6.4 Conclusions

We present the design and implementation of a novel flexible virtual fixture interface. By allowing users to visually specify path constraints for a tele-manipulation task we were able to provide a system where performance of unilateral (gamepad - WAM) configurations began to match that of a bilateral (WAM - WAM) system. The lack of feedback in unilateral control was compensated through our interface. In our pilot user study smoothness (SAL metric) was improved by a factor of 2 over all the tasks performed by users when using our interface. Similarly users were able to reduce the task time by a factor of 2.1. Reducing the control degrees of freedom, results in reducing their perceived workload. The users not only prefer to use our teleop interface, but their performance is also improved. One area of future work is the

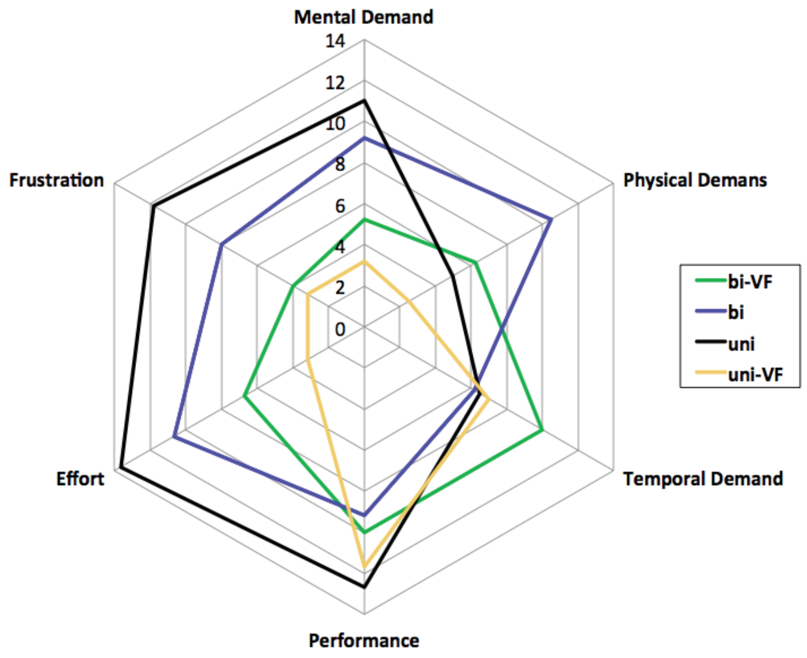


Figure 6.13: Nasa TLX radar plot: Qualitative workload evaluation, higher workloads are farther from the center of plot.

study of the effects of calibration error on our system. When users try to fight the forces introduced by the system, we see a serious increase in the trajectory error. User's attention is more focused on trying to correct for the calibration offset rather than on completing the task which results in performance degradation and increase in mental demand.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The robotics field is facing an exciting transition, where robots are appearing more often in human environments. However, one of the main challenges in this transition is the lack of natural communication mechanisms between humans and robots.

In this thesis, we address this problem by studying human pointing as a human-robot communication mechanism. We have proposed three novel interfaces: Spatial Pointing (Chapter 4), Display-based pointing (Chapter 5) and Remote Display-based pointing (Chapter 6).

In the spatial pointing case (Chapter 4), we have found that by developing an interface capable of interpreting and interacting through human pointing, the robot can simplify and solve complex object recognition and localization problems. Furthermore, by communicating through spatial gestures, the robot can leverage human knowledge to solve current AI and Computer Vision problems (see the case study in Chapter 4.5).

A similar result was obtained in Chapter 5 during the experiments with the proposed 2D upper-body disabled interface, where the user decides the grasping orientation, while avoiding possible collisions. Another meaningful result in Chapter 5, was the reduction in the number of DOF between the proposed interface and the current way of operating the assistive robot (joystick device, see Chapter 2.1.6). The proposed interface design reduced the multiple DOF control and mode shifting to 1 or 2 DOF control.

In Chapter 6 we introduced a remote path specification interface using vision and force control. Similar to Chapter 5, we reduced the require DOF of operation from 6

DOF to 1DOF. Furthermore, through the developed interface, the experimental results showed that users performed equally well using a bidirectional and unidirectional configuration.

7.2 Future Work

Here we present some of our ongoing work and future ideas for the different developed interfaces.

7.2.1 Spatial Pointing

To integrate robots into human environments it is necessary to provide natural communication mechanisms as proposed in this thesis. As shown in the different case studies, a good interface can solve complex AI and Computer Vision problems by relying on human knowledge. However, in the long-term, building learning mechanisms into the robot is the ultimate goal.

Two recent technologies that could leverage this objective are Deep Learning [127] and Cloud Robotics. "Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction"[128]. It is a powerful tool that has improved the state-of-the-art of many domains, e.g., object detection, object recognition, speech recognition, etc. The Cloud Robotics paradigm consists of sharing other robot experience information through a network. In Cloud Robotics the robot can learn general actions from other robots, of course this is subject to many technical details, e.g., different types of robots, sensors, environments, etc.

In our proposed future work we are assuming that the robot has a general knowledge of the world. This general knowledge can come from the network or a factory pre-learned module. We propose to study communication channels to introduce or refine robot knowledge through HRI. Similar to the idea of cloud computing, with the difference that the information comes from interacting or seeing human interacting in the same robot environment. Our plan is to use our spatial pointing module as a centerpiece of the interaction. Below we provide a case study and the current state of our system.

7.2.1.1 Interaction example

An interaction example using our proposed interface is shown in Fig. 7.1. Our example shows a person soldering a circuit board. During this work, he requires the use of a multimeter, that is out of his reach. He asks his robot assistant to bring the multimeter. (Fig. 7.1A).

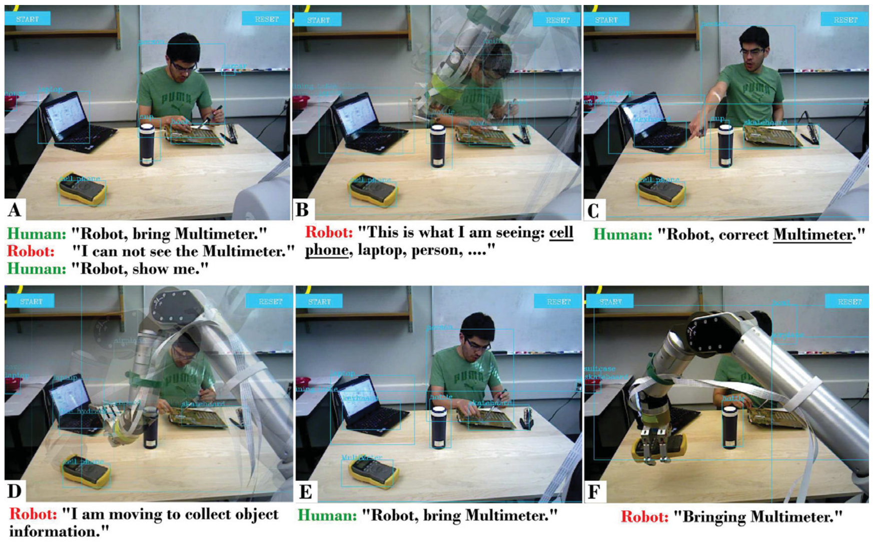

A
Human: "Robot, bring Multimeter."
Robot: "I can not see the Multimeter."
Human: "Robot, show me."
B
Robot: "This is what I am seeing: cell phone, laptop, person, ..."
C
Human: "Robot, correct Multimeter."
D
Robot: "I am moving to collect object information."
E
Human: "Robot, bring Multimeter."
F
Robot: "Bringing Multimeter."

Figure 7.1: **Incrementing robot knowledge through HRI:**
A) The human ask to bring the multimeter while working on a circuit board. The robot does not know what a multimeter is. The human asks for the robot's world representation.
B) The robot iterates through the detected objects by pointing and saying each object label.
C) The human points and corrects the "multimeter" label which was initially recognize as a "cell phone".
D) The robot goes close to the pointed object and collects images of the corrected object.
E,F) The human ask again to bring the multimeter. This time the robot succeeds in his task.

The robot is equipped with a recognition module. Unfortunately, the multimeter class was not found as a recognizable object. The person then asks the robot what it sees (given its current world representation) (Fig. 7.1A), to figure out if the robot is detecting the object under another name/category or it does not see it at all.
Through speech, the robot communicates the recognized objects in the scene, and by using pointing gestures the robot provides object's locations (Fig. 7.1B). Pointing

allows the robot to skip complex spatial sentences, (e.g. "from my point of view the laptop is on the left top corner of the table"). After communicating which objects were recognized and localized by the robot the human can interact through gestures and speech to correct or add a particular object (Fig. 7.1C), in this case, the multimeter. After the addition or correction, the robot collects several images on-line of the target object (Fig. 7.1D) which is used to modify the current world representation. This procedure needs to be done only once for each new class. Finally, the person asks again to bring the multimeter (Fig. 7.1E). This time the robot succeeds on recognizing the object and executes a picking task to bring the multimeter. (Fig. 7.1F).

7.2.1.2 Current System

Our proposed system uses a 7-DOF WAM arm instrumented with eye-in-hand-cameras, a microphone, Kinect camera and speakers. It is composed of 7 modules as shown in Fig. 7.2, all modules are fully integrated with ROS [103].

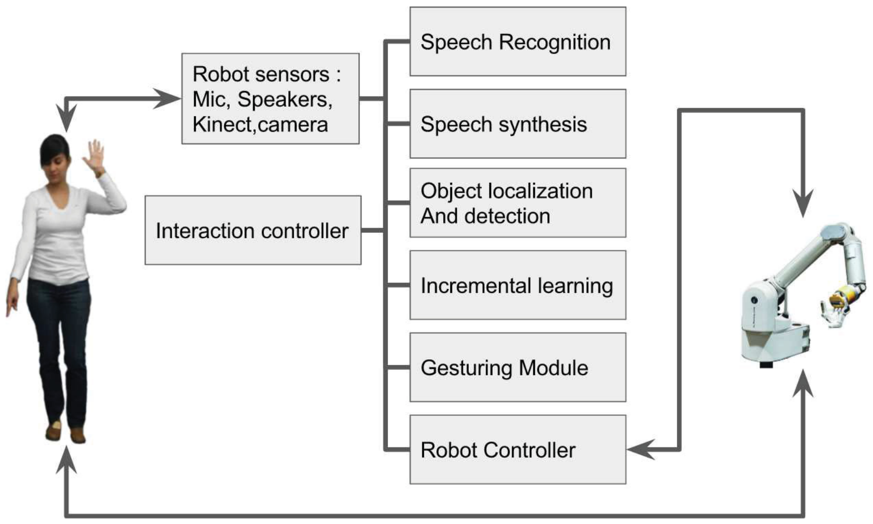


Figure 7.2: System block diagram.

- The **speech recognition module** integrates the CMU Sphinx toolkit [129]. It provides basic word and sentence recognition used by the robot to shift states during the interaction.

- The **speech synthesis module** relies on the Festival speech synthesis system [130] and provides feedback to the human in a verbal channel.

- The **object localization and detection module** provides labels and 2D locations of the objects in the scene. For a detail description see our preprint paper version [131].

- **The Incremental learning module** uses HRI, to permit changes in the robot's world representation. For a detail description see our preprint paper version [131].

- The **gesturing module** is based on this thesis work.

- The **robot controller module** commands robot movements and generates: pointing gestures, robot data collection and pick-up object actions.

- The **interaction controller module** is in charge of orchestrating the complete system. It supports the different interactions that are shown in Fig. 7.1 and it is based on a finite state machine that is triggered by gesture or speech coming from the human and/or robot.

In Fig. 7.1 four important interactions of our system are highlighted. **Verbal interaction**, by using both speech recognition and synthesis the human and the robot can establish basic verbal communication. The **ground truth world representation interaction** presents both verbal and gesture interaction performed by the robot. The recognition and localization module provides 2D bounding boxes and labels of the detected objects. The system verbally informs the object class and at the same time the robot arm points to the object 3D centroid. The centroid is calculated by using the RGB to depth camera correspondence from the objects bounding box (See Fig. 7.3).

In the **correct interaction**, human uses both verbal and gesture communication to annotate a particular object in the scene that needs to be corrected. Fig 7.3 shows both RGB and Point cloud visualization. The head and hand of the human are tracked as two points. With a verbal triggering, a 3D ray is constructed from these two points, hitting the target object. The robot is then commanded to collect data with this 3D object location. The object collection is performed through the eye-in-hand camera by moving the end-effector in a parameterized helix curve keeping the

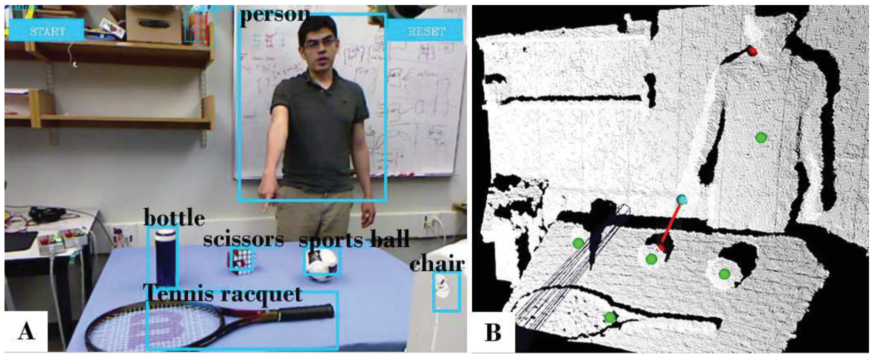


Figure 7.3: A) RGB visualization. Objects in the scene are detected and localized. The human points to the rubik's cube to correct its label. *Note: the font and the line-width of the bounding boxes are enlarged from the original image for clarity.* B) Point cloud visualization. Using the 2D to 3D correspondence, 2D bounding boxes centroids are used to find 3D objects centroids (green spheres).

camera facing to the object location. During the collection state, a TLD tracker [132] is used to guarantee the cropping of the object during the data collection. The initial bounding box of the tracker is given as the whole image when the robots starts the collection close to the object.

7.2.2 Proximate Display-based Pointing

In Chapter 5.4 we provide a detail discussion and improvements for this work. Also, we believe our current prototype is ready to be tested with an end user group. We have been in contact with researchers at the Glenrose Hospital, and we have presented our work in the "I Can Centre" (A center to promote assistive technologies). We have received positive feedback from clinicians. Our next step is to find a researcher partner, working on upper body disable assistance that would help us performing trials with end users.

7.2.3 Remote Display-based Pointing

So far in the remote display-based we have only defined a single path. A possible improvement is to introduce multiple paths. Figure 7.4 shows a cake cutting mock-up task. The task consist of performing an initial round cut in the middle (blue path) and then a slice cut (red path). Multiple paths specifications enables to break down a complex task into several simple sub-tasks. In addition for the tasks that require high precision, we are planning to include two eye-in-hand cameras and develop a hybrid image based visual servoing [133] combined with our current implementation to improve precision. This is in the same line of the idea presented in Chapter 5.4.2

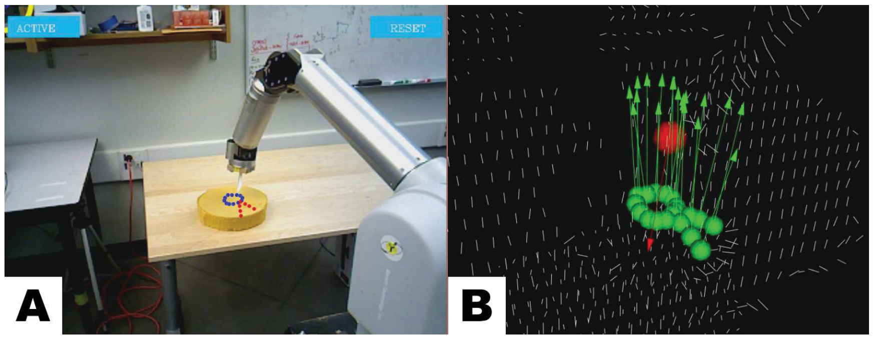


Figure 7.4: A) A circular path (blue dots) and a triangular path (red dots) are defined; the user can switch between both the paths to complete the cutting task. B) The scene point cloud is visualized with both paths (green spheres) and the calculated normals (green arrows).

Bibliography

[1] M. Ghodoussi, S. E. Butner, and Y. Wang, “Robotic surgery-the transatlantic case,” in *Robotics and Automation, 2002. Proceedings. ICRA’02. IEEE International Conference on*, vol. 2. IEEE, 2002, pp. 1882–1888.

[2] T. B. Sheridan and W. L. Verplank, “Human and computer control of undersea teleoperators,” DTIC Document, Tech. Rep., 1978.

[3] T. L. Chen, M. Ciocarlie, S. Cousins, P. Grice, K. Hawkins, K. Hsiao, C. C. Kemp, C.-H. King, D. A. Lazewatsky, A. E. Leeper *et al.*, “Robots for humanity: Using assistive robots to empower people with disabilities,” 2013.

[4] S. S. Rautaray and A. Agrawal, “Vision based hand gesture recognition for human computer interaction: a survey,” *Artificial Intelligence Review*, vol. 43, no. 1, pp. 1–54, 2015.

[5] M. Land, N. Mennie, J. Rusted *et al.*, “The roles of vision and eye movements in the control of activities of daily living,” *Perception-London*, vol. 28, no. 11, pp. 1311–1328, 1999.

[6] M. M. Hayhoe, A. Shrivastava, R. Mruczek, and J. B. Pelz, “Visual memory and motor planning in a natural task,” *Journal of vision*, vol. 3, no. 1, p. 6, 2003.

[7] K. M. Tsui, D.-J. Kim, A. Behal, D. Kontak, and H. A. Yanco, “i want that: Human-in-the-loop control of a wheelchair-mounted robotic arm,” *Applied Bionics and Biomechanics*, vol. 8, no. 1, pp. 127–147, 2011.

[8] A. E. Leeper, K. Hsiao, M. Ciocarlie, L. Takayama, and D. Gossow, “Strategies for human-in-the-loop robotic grasping,” in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction.* ACM, 2012, pp. 1–8.

[9] N. Wiener, *God and golem, Inc.: a comment on certain points where cybernetics impinges on religion.* MIt Press, 1966.

[10] E. Papadopoulos, "Heron of alexandria (c. 10–85 ad)," in *Distinguished figures in mechanism and machine science.* Springer, 2007, pp. 217–245.

[11] M. E. Rosheim, "Leonardos knight," *Leonardos Lost Robots*, pp. 69–113, 2006.

[12] S. S. Srinivasa, D. Berenson, M. Cakmak, A. Collet, M. R. Dogar, A. D. Dragan, R. A. Knepper, T. Niemueller, K. Strabala, M. Vande Weghe *et al.*, "Herb 2.0: Lessons learned from developing a mobile manipulator for the home," *Proceedings of the IEEE*, vol. 100, no. 8, pp. 2410–2428, 2012.

[13] G. A. Pratt, "Robot to the rescue," *Bulletin of the Atomic Scientists*, vol. 70, no. 1, pp. 63–69, 2014.

[14] C. C. Kemp, A. Edsinger, and E. Torres-Jara, "Challenges for robot manipulation in human environments," *Robotics & Automation Magazine, IEEE*, vol. 14, no. 1, pp. 20–29, 2007.

[15] M. A. Goodrich and A. C. Schultz, "Human-robot interaction: a survey," *Foundations and trends in human-computer interaction*, vol. 1, no. 3, pp. 203–275, 2007.

[16] M. Soegaard and R. F. Dam, *Encyclopedia of Human-Computer Interaction.* Interaction Design Foundation, 2013.

[17] J. G. Trafton, N. L. Cassimatis, M. D. Bugajska, D. P. Brock, F. E. Mintz, and A. C. Schultz, "Enabling effective human-robot interaction using perspective-taking in robots," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 35, no. 4, pp. 460–470, 2005.

[18] B. Gleeson, K. MacLean, A. Haddadi, E. Croft, and J. Alcazar, "Gestures for industry: intuitive human-robot communication from human observation," in *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction.* IEEE Press, 2013, pp. 349–356.

[19] G. Gibbs and S. Sachdev, "Canada and the international space station program: overview and status," *Acta Astronautica*, vol. 51, no. 1, pp. 591–600, 2002.

[20] W. T. Townsend and J. K. Salisbury, "Mechanical design for whole-arm manipulation," in *Robots and Biological Systems: Towards a New Bionics?* Springer, 1993, pp. 153–164.

[21] C. Loughlin, A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, and G. Hirzinger, "The dlr lightweight robot: design and control concepts for robots in human environments," *Industrial Robot: an international journal*, vol. 34, no. 5, pp. 376–385, 2007.

[22] V. Maheu, J. Frappier, P. Archambault, and F. Routhier, "Evaluation of the jaco robotic arm: Clinico-economic study for powered wheelchair users with upper-extremity disabilities," in *Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on.* IEEE, 2011, pp. 1–5.

[23] S. Haddadin and E. Croft, "Physical human–robot interaction," in *Springer Handbook of Robotics.* Springer, 2016, pp. 1835–1874.

[24] C. Heyer, "Human-robot interaction and future industrial robotics applications." in *IROS*, 2010, pp. 4749–4754.

[25] M. Van der Loos, D. Reinkensmeyer, and E. Guglielmelli, "Rehabilitation and health care robotics," in *Springer Handbook of Robotics.* Springer, 2016, pp. 1685–1728.

[26] K. Yoshida, D. Nenchev, G. Ishigami, and Y. Tsumaki, "Space robotics," in *The International Handbook of Space Technology.* Springer, 2014, pp. 541–573.

[27] R. R. Murphy, *Disaster robotics.* MIT press, 2014.

[28] E. Prassler, M. Munich, P. Pirjanian, and K. Kosuge, "Domestic robotics," in *Springer Handbook of Robotics.* Springer, 2016, pp. 1729–1758.

[29] M. Bergerman, J. Billingsley, J. Reid, and E. van Henten, "Robotics in agriculture and forestry," in *Springer Handbook of Robotics.* Springer, 2016, pp. 1463–1492.

[30] H.-T. Choi and J. Yuh, "Underwater robots," in *Springer Handbook of Robotics.* Springer, 2016, pp. 595–622.

[31] B. Driessen, T. T. Kate, F. Liefhebber, A. Versluis, and J. van Woerden, “Collaborative control of the manus manipulator,” *Universal Access in the Information Society*, vol. 4, no. 2, pp. 165–173, 2005.

[32] G. W. Römer, H. Stuyt, G. Peters, and K. van Woerden, “14 processes for obtaining a manus(arm) robot within the netherlands,” in *Advances in Rehabilitation Robotics*. Springer, 2004, pp. 221–230.

[33] R. S. Woodworth, “Accuracy of voluntary movement.” *The Psychological Review: Monograph Supplements*, vol. 3, no. 3, p. i, 1899.

[34] K. M. Tsui and H. A. Yanco, “Simplifying wheelchair mounted robotic arm control with a visual interface.” in *AAAI Spring Symposium: Multidisciplinary Collaboration for Socially Assistive Robotics*, 2007, pp. 97–102.

[35] K. Tsui, H. Yanco, D. Kontak, and L. Beliveau, “Development and evaluation of a flexible interface for a wheelchair mounted robotic arm,” in *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*. ACM, 2008, pp. 105–112.

[36] C. Dune, E. Marchand, C. Collowet, and C. Leroux, “Active rough shape estimation of unknown objects,” in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 3622–3627.

[37] C. Dune, A. Remazeilles, E. Marchand, C. Leroux *et al.*, “Vision-based grasping of unknown objects to improve disabled people autonomy.” in *Robotics: Science and Systems Manipulation Workshop: Intelligence in Human Environments.*, 2008.

[38] U. Reiser, T. Jacobs, G. Arbeiter, C. Parlitz, and K. Dautenhahn, “Care-o-bot® 3–vision of a robot butler,” in *Your virtual butler*. Springer, 2013, pp. 97–116.

[39] A. Jain and C. C. Kemp, “El-e: an assistive mobile manipulator that autonomously fetches objects from flat surfaces,” *Autonomous Robots*, vol. 28, no. 1, pp. 45–64, 2010.

[40] A. Remazeilles, C. Leroux, and G. Chalubert, “Sam: a robotic butler for handicapped people,” in *Robot and Human Interactive Communication, 2008. RO-*

MAN 2008. The 17th IEEE International Symposium on. IEEE, 2008, pp. 315–321.

[41] M. Ciocarlie, K. Hsiao, A. Leeper, and D. Gossow, "Mobile manipulation through an assistive home robot," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on.* IEEE, 2012, pp. 5313–5320.

[42] J. Eisenstein and R. Davis, "Visual and linguistic information in gesture classification," in *Proceedings of the 6th international conference on Multimodal interfaces.* ACM, 2004, pp. 113–120.

[43] S. Mitra and T. Acharya, "Gesture recognition: A survey," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 37, no. 3, pp. 311–324, 2007.

[44] S. Kita, *Pointing: Where language, culture, and cognition meet.* Psychology Press, 2003.

[45] G. Butterworth, F. Simion *et al.*, *The Development Of Sensory, Motor And Cognitive Capacities In Early Infancy: From Sensation To Cognition.* Psychology Press, 2013.

[46] M. Tomasello, *Origins of human communication.* MIT press Cambridge, 2008.

[47] S. C. Lozano and B. Tversky, "Communicative gestures facilitate problem solving for both communicators and recipients," *Journal of Memory and Language*, vol. 55, no. 1, pp. 47–63, 2006.

[48] R. E. Kahn and M. J. Swain, "Understanding people pointing: The perseus system," in *International Symposium on Computer Vision.* Citeseer, 1995, pp. 569–574.

[49] N. Jojic, B. Brumitt, B. Meyers, S. Harris, and T. Huang, "Detection and estimation of pointing gestures in dense disparity maps," in *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on.* IEEE, 2000, pp. 468–475.

[50] K. Nickel and R. Stiefelhagen, "Pointing gesture recognition based on 3d-tracking of face, hands and head orientation," in *Proceedings of the 5th international conference on Multimodal interfaces.* ACM, 2003, pp. 140–146.

[51] R. Kehl and L. Van Gool, "Real-time pointing gesture recognition for an immersive environment," in *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on.* IEEE, 2004, pp. 577–582.

[52] M. Pateraki, H. Baltzakis, and P. Trahanias, "Visual estimation of pointed targets for robot guidance via fusion of face pose and hand orientation," *Computer Vision and Image Understanding*, 2013.

[53] S. Waldherr, R. Romero, and S. Thrun, "A gesture based interface for human-robot interaction," *Autonomous Robots*, vol. 9, no. 2, pp. 151–173, 2000.

[54] D. Droeschel, J. Stückler, and S. Behnke, "Learning to interpret pointing gestures with a time-of-flight camera," in *Proceedings of the 6th international conference on Human-robot interaction.* ACM, 2011, pp. 481–488.

[55] M. Van den Bergh, D. Carton, R. De Nijs, N. Mitsou, C. Landsiedel, K. Kuehnlenz, D. Wollherr, L. Van Gool, and M. Buss, "Real-time 3d hand gesture interaction with a robot for understanding directions from humans," in *RO-MAN, 2011 IEEE.* IEEE, 2011, pp. 357–362.

[56] C. C. Kemp, C. D. Anderson, H. Nguyen, A. J. Trevor, and Z. Xu, "A point-and-click interface for the real world: Laser designation of objects for mobile manipulation," in *Proc. 3rd ACM/IEEE Int Human-Robot Interaction (HRI) Conf*, 2008, pp. 241–248.

[57] M.-A. Williams, S. Abidi, P. Gärdenfors, X. Wang, B. Kuipers, and B. Johnston, "Interpreting robot pointing behavior," in *Social Robotics.* Springer, 2013, pp. 148–159.

[58] C.-M. Huang and B. Mutlu, "Modeling and evaluating narrative gestures for humanlike robots," *Proceedings of the 9th R: SS*, 2013.

[59] T. Ende, S. Haddadin, S. Parusel, T. Wusthoff, M. Hassenzahl, and A. Albu-Schaffer, "A human-centered approach to robot gesture based communication within collaborative working processes," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on.* IEEE, 2011, pp. 3367–3374.

[60] C. Rich, B. Ponsler, A. Holroyd, and C. L. Sidner, “Recognizing engagement in human-robot interaction,” in *Human-Robot Interaction (HRI), 2010 5th ACM/IEEE International Conference on.* IEEE, 2010, pp. 375–382.

[61] D. A. Bowman and L. F. Hodges, “An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments,” in *Proceedings of the 1997 symposium on Interactive 3D graphics.* ACM, 1997, pp. 35–ff.

[62] M. R. Mine, F. P. Brooks Jr, and C. H. Sequin, “Moving objects in space: exploiting proprioception in virtual-environment interaction,” in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques.* ACM Press/Addison-Wesley Publishing Co., 1997, pp. 19–26.

[63] J. Liang and M. Green, “Jdcad: A highly interactive 3d modeling system,” *Computers & graphics*, vol. 18, no. 4, pp. 499–506, 1994.

[64] C.-B. Park and S.-W. Lee, “Real-time 3d pointing gesture recognition for mobile robots with cascade hmm and particle filter,” *Image and Vision Computing*, vol. 29, no. 1, pp. 51–63, 2011.

[65] S. Hutchinson, G. D. Hager, and P. I. Corke, “A tutorial on visual servo control,” *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 5, pp. 651–670, 1996.

[66] A. Sanderson and L. Weiss, “Image-based visual servo control using relational graph error signals,” *Proc. ieee*, vol. 1074, 1980.

[67] E. Malis, F. Chaumette, and S. Boudet, “Positioning a coarse-calibrated camera with respect to an unknown object by 2d 1/2 visual servoing,” in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 2. IEEE, 1998, pp. 1352–1359.

[68] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision.* Cambridge university press, 2003.

[69] F. Chaumette and S. Hutchinson, “Visual servo control. i. basic approaches,” *Robotics & Automation Magazine, IEEE*, vol. 13, no. 4, pp. 82–90, 2006.

[70] F. Chaumette, S. Hutchinson *et al.*, "Visual servo control, part ii: Advanced approaches," *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 109–118, 2007.

[71] D. Kragic, H. I. Christensen *et al.*, "Survey on visual servoing for manipulation," *Computational Vision and Active Perception Laboratory, Fiskartorpsv*, vol. 15, 2002.

[72] R. T. Fomena, C. Perez Quintero, M. Gridseth, and M. Jagersand, "Towards practical visual servoing in robotics," in *Computer and Robot Vision (CRV), 2013 International Conference on.* IEEE, 2013, pp. 303–310.

[73] M. Gridseth, C. Quintero, R. T. Fomena, O. Ramirez, and M. Jagersand, "Bringing visual servoing into real world applications," in *Human Robot Collaboration Workshop, Robotics Science and Systems RSS*, vol. 13.

[74] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[75] A. Farahmand, A. Shademan, and M. Jagersand, "Global visual-motor estimation for uncalibrated visual servoing," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on.* IEEE, 2007, pp. 1969–1974.

[76] M. Jagersand, O. Fuentes, and R. Nelson, "Experimental evaluation of uncalibrated visual servoing for precision manipulation," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 4. IEEE, 1997, pp. 2874–2880.

[77] J. A. Piepmeier, G. V. McMurray, and H. Lipkin, "Uncalibrated dynamic visual servoing," *Robotics and Automation, IEEE Transactions on*, vol. 20, no. 1, pp. 143–147, 2004.

[78] H. Sutanto, R. Sharma, and V. Varma, "The role of exploratory movement in visual servoing without calibration," *Robotics and Autonomous Systems*, vol. 23, no. 3, pp. 153–169, 1998.

[79] K. Hosoda and M. Asada, "Versatile visual servoing without knowledge of true jacobian," in *Intelligent Robots and Systems' 94.'Advanced Robotic Systems*

and the Real World', IROS'94. Proceedings of the IEEE/RSJ/GI International Conference on, vol. 1. IEEE, 1994, pp. 186–193.

[80] J. P. Hespanha, Z. Dodds, G. D. Hager, and A. S. Morse, "What tasks can be performed with an uncalibrated stereo vision system," *International Journal on Computer Vision*, vol. 35, pp. 65–85, 1999.

[81] Z. Dodds, G. D. Hager, A. S. Morse, and J. P. Hespanha, "Task specification and monitoring for uncalibrated hand/eye coordination," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2. IEEE, 1999, pp. 1607–1613.

[82] R. B. Rusu, *Semantic 3D object maps for everyday robot manipulation.* Springer, 2013.

[83] K. Hsiao, S. Chitta, M. Ciocarlie, and E. G. Jones, "Contact-reactive grasping of objects with partial shape information," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on.* IEEE, 2010, pp. 1228–1235.

[84] C. Matuszek, A. Pronobis, L. Zettlemoyer, and D. Fox, "Combining world and interaction models for human-robot collaborations," in *Proceedings of Workshops at the 27th AAAI Conference on Artificial Intelligence (AAAI 2013), Bellevue, Washington, DC, USA*, 2013, pp. 14–15.

[85] 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 *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop*, Kobe, Japan, may 2009.

[86] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on.* IEEE, 2011, pp. 1–4.

[87] G. Bradski, *Dr. Dobb's Journal of Software Tools.*

[88] *OpenNI User Guide*, OpenNI Organization, August 2012, last viewed 23-08-2012. [Online]. Available: http://www.openni.org/documentation

[89] C. Perez Quintero, R. Tatsambon Fomena, A. Shademan, N. Wolleb, T. Dick, and M. Jagersand, “Sepo: Selecting by pointing as an intuitive human-robot command interface,” in *Robotics and Automation(ICRA)*, 2013.

[90] C. Herrera, J. Kannala, J. Heikkilä *et al.*, “Joint depth and color camera calibration with distortion correction,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 10, pp. 2058–2064, 2012.

[91] M. Fischler and R. Bolles, “Random sample consensus,” *Communications of the ACM*, vol. 24, no. 6, 1981.

[92] H. Nguyen, M. Ciocarlie, K. Hsiao, and C. C. Kemp, “Ros commander (rosco): Behavior creation for home robots,” in *Robotics and Automation (ICRA), 2013 IEEE International Conference on.* IEEE, 2013, pp. 467–474.

[93] K. Nickel, E. Scemann, and R. Stiefelhagen, “3d-tracking of head and hands for pointing gesture recognition in a human-robot interaction scenario,” in *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on.* IEEE, 2004, pp. 565–570.

[94] C.-S. Chung, H. Wang, and R. A. Cooper, “Functional assessment and performance evaluation for assistive robotic manipulators: Literature review,” *The Journal of Spinal Cord Medicine*, vol. 36, no. 4, pp. 273–289, 2013.

[95] Kinova, “Jaco usability study results,” in *Feb report.* Robotics, 2014, pp. 303–310.

[96] D.-J. Kim, R. Hazlett-Knudsen, H. Culver-Godfrey, G. Rucks, T. Cunningham, D. Portee, J. Bricout, Z. Wang, and A. Behal, “How autonomy impacts performance and satisfaction: Results from a study with spinal cord injured subjects using an assistive robot,” *Systems, Man and Cybernetics, IEEE*, vol. 42, no. 1, pp. 2–14, 2012.

[97] C. Perez Quintero, O. Ramirez, and M. Jagersand, “Vibi: Assistive vision-based interface for robot manipulation,” in *Robotics and Automation (ICRA), 2015 IEEE International Conference on.* IEEE, 2015, p. To appear in.

[98] Y. Matsumoto, Y. Nishida, Y. Motomura, and Y. Okawa, “A concept of needs-oriented design and evaluation of assistive robots based on icf,” in *2011 IEEE International Conference on Rehabilitation Robotics.* IEEE, 2011, pp. 1–6.

[99] R. Rusu, "3d is here:point cloud library (pcl)," in *Robotics and Automation(ICRA)*, 2011.

[100] C. P. Quintero, M. Dehghan, O. Ramirez, M. H. Ang, and M. Jagersand, "Vision-force interface for path specification in tele-manipulation," in *Human-Robot Interfaces for Enhanced Physical Interactions Workshop at ICRA 2016*, May 2016.

[101] S. Holzer, R. B. Rusu, M. Dixon, S. Gedikli, and N. Navab, "Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on.* IEEE, 2012, pp. 2684–2689.

[102] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke, "Real-time plane segmentation using rgb-d cameras," in *RoboCup 2011: robot soccer world cup XV.* Springer, 2011, pp. 306–317.

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

[104] S. Chitta, I. Sucan, and S. Cousins, "Moveit!" *IEEE Robotics Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[105] https://webdocs.cs.ualberta.ca/~vis/HRI/VIBI_conf.mp4, June 2016.

[106] https://webdocs.cs.ualberta.ca/~vis/HRI/VIBI_comp.wmv, June 2016.

[107] M. Gridseth, O. Ramirez, C. P. Quintero, and M. Jagersand, "Vita: Visual task specification interface for manipulation with uncalibrated visual servoing," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 3434–3440.

[108] H. A. Yanco, A. Norton, W. Ober, D. Shane, A. Skinner, and J. Vice, "Analysis of human-robot interaction at the darpa robotics challenge trials," *Journal of Field Robotics*, vol. 32, no. 3, pp. 420–444, 2015.

[109] Y. Mezouar, M. Prats, and P. Martinet, "External hybrid vision/force control," in *Intl. Conference on Advanced Robotics (ICAR07)*, 2007.

[110] G. Morel, E. Malis, and S. Boudet, "Impedance based combination of visual and force control," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 2. IEEE, 1998, pp. 1743–1748.

[111] K. Hosoda, K. Igarashi, and M. Asada, "Adaptive hybrid visual servoing/force control in unknown environment," in *Intelligent Robots and Systems' 96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, vol. 3. IEEE, 1996, pp. 1097–1103.

[112] A. Pichler and M. Jagersand, "Uncalibrated hybrid force-vision manipulation," in *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2000, pp. 1866–1871.

[113] S. A. Bowyer, B. L. Davies, and F. Rodriguez y Baena, "Active constraints/virtual fixtures: A survey," *Robotics, IEEE Transactions on*, vol. 30, no. 1, pp. 138–157, 2014.

[114] A. Bettini, P. Marayong, S. Lang, A. M. Okamura, and G. D. Hager, "Vision-assisted control for manipulation using virtual fixtures," *IEEE Transactions on Robotics*, vol. 20, no. 6, pp. 953–966, Dec 2004.

[115] A. Y. C. Nee, *Handbook of Manufacturing Engineering and Technology*. Springer Publishing Company, Incorporated, 2014.

[116] D. Nakhaeinia, R. Fareh, P. Payeur, and R. Laganiere, "Trajectory planning for surface following with a manipulator under rgb-d visual guidance," in *Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on*, Oct 2013, pp. 1–6.

[117] A. M. Okamura, "Methods for haptic feedback in teleoperated robot-assisted surgery," *Industrial Robot: An International Journal*, vol. 31, no. 6, pp. 499–508, 2004.

[118] T. Yamamoto, N. Abolhassani, S. Jung, A. M. Okamura, and T. N. Judkins, "Augmented reality and haptic interfaces for robot-assisted surgery," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 8, no. 1, pp. 45–56, 2012. [Online]. Available: http://dx.doi.org/10.1002/rcs.421

[119] Z. Pezzementi, A. M. Okamura, and G. D. Hager, "Dynamic guidance with pseudoadmittance virtual fixtures," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 1761–1767.

[120] R. A. Castillo-Cruces and J. Wahrburg, "Virtual fixtures with autonomous error compensation for human-robot cooperative tasks," *Robotica*, vol. 28, no. 2, pp. 267–277, 009 2009.

[121] S. Vozar, S. Leonard, P. Kazanzides, and L. Whitcomb, "Experimental evaluation of force control for virtual-fixture-assisted teleoperation for on-orbit manipulation of satellite thermal blanket insulation," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, May 2015, pp. 4424–4431.

[122] T. Xia, S. Leonard, A. Deguet, L. Whitcomb, and P. Kazanzides, "Augmented reality environment with virtual fixtures for robotic telemanipulation in space," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, Oct 2012, pp. 5059–5064.

[123] T. Xia, S. Leonard, I. Kandaswamy, A. Blank, L. Whitcomb, and P. Kazanzides, "Model-based telerobotic control with virtual fixtures for satellite servicing tasks," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, May 2013, pp. 1479–1484.

[124] Z. Jiang, Y. Liu, H. Liu, and J. Zou, "Flexible virtual fixture enhanced by vision and haptics for unstructured environment teleoperation," in *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec 2013, pp. 2643–2648.

[125] C. Glover, B. Russell, A. White, M. Miller, and A. Stoytchev, "An effective and intuitive control interface for remote robot teleoperation with complete haptic feedback," in *proceedings of the Emerging Technologies Conference-ETC*, 2009.

[126] S. Balasubramanian, A. Melendez-Calderon, and E. Burdet, "A robust and sensitive metric for quantifying movement smoothness," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 8, pp. 2126–2136, Aug 2012.

[127] G. A. Pratt, "Is a cambrian explosion coming for robotics?" *The Journal of Economic Perspectives*, vol. 29, no. 3, pp. 51–60, 2015.

[128] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[129] "CMU Sphinx Open Source Toolkit for Speech Recognition project by carnegie mellon university," http://cmusphinx.sourceforge.net/, accessed: 2016-06-25.

[130] A. W. Black and P. A. Taylor, "The Festival Speech Synthesis System: System documentation," Human Communciation Research Centre, University of Edinburgh, Scotland, UK, Tech. Rep. HCRC/TR-83, 1997, avaliable at http://www.cstr.ed.ac.uk/projects/festival.html.

[131] S. Valipour, C. Perez, and M. Jagersand, "Incremental learning for robot perception through hri," *arXiv preprint arXiv:1701.04693*, 2017.

[132] Z. Kalal, J. Matas, and K. Mikolajczyk, "Pn learning: Bootstrapping binary classifiers by structural constraints," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on.* IEEE, 2010, pp. 49–56.

[133] C. P. Quintero, O. Ramirez, M. Gridseth, and M. Jägersand, "Small object manipulation in 3d perception robotic systems using visual servoing." IROS Workshop, 2014.