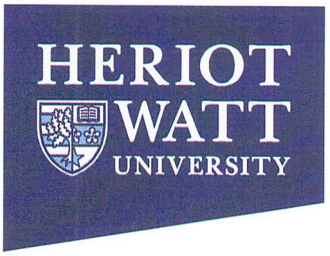
Heriot-Watt University

School of Engineering and Physical Sciences

MSc in Robotics, Autonomous and Interactive Systems specialising in Marine Robotics

Dissertation

Title:	**Gesture Control of an Underwater Manipulator**
Author:	**Roshenac Mitchell**
Registration Number	**H00074473**
Date	**26 August 2016**
Supervisor:	**Dr Matthew Dunnigan and Prof. David Lane**



Declaration of authorship

I, Roshenac Mitchell

confirm that the report entitled Gesture Control of an Underwater Manipulator is part of my assessment for module Robotics, Autonomous and Interactive Systems specialising in Marine Robotics

I declare that the report is my own work. I have not copied other material verbatim except in explicit quotes, and I have identified the sources of the material clearly.

R. Mitchell ..... Heriot-Watt Edinburgh, 26 August 2016

(Signature)

Heriot-Watt University

Masters Thesis

Gesture Control of an Underwater Manipulator

Author:

Roshenac Mitchell

Enrolment number:

H00074473

Supervisors:

Dr. Matthew Dunnigan

Prof. David Lane

Dissertation submitted in partial fulfilment of the requirements for the degree of MSc at Heriot-Watt University

in the

School of Engineering and Physical Sciences

November 8, 2016

Abstract

Current teleoperation controllers, such as operator control units, are difficult to use without operators going through extensive training. This research tests the hypothesis that a gesture control input device would provide a more intuitive method of controlling the HDT Adroit Underwater Manipulator. This manipulator is currently controlled using a joystick based Hardened Operator Control Unit (HOCU). To test this hypothesis this research uses a Leap Motion input device to track an operator's hand. Inverse kinematics is then used to calculate the required joint angles which are needed to manoeuvre the manipulator's end effector to the required position.

The findings of this research are based on a number of timed task trials and a questionnaire. These results suggest that the Leap Motion device is more capable, intuitive and easier to control, compared to the HOCU. This implies that gesture controllers, such as the Leap Motion device, are a plausible replacement for a teleoperation based control input. It is further suggested that the teleoperation control of the manipulator can advanced with the addition of virtual reality and haptic feedback..

Keywords: *manipulator, Leap Motion, teleoperation, underwater*

Acknowledgements

I would like to thank Matthew Dunnigan for taking the time to discuss ideas with me. I would also like to thank Bence Magyar and Corina Barbalata who have pointed me in the right direction and always answered any questions I had. Lastly, I would like to thank my family for always encouraging me to think bigger and proofreading for me when I needed it.

Contents

List of Figures

Abbreviations

DNC	**D**id **N**ot (manage to) **C**omplete (task due to connection fault)
DNT	**D**id **N**ot **T**ime (this task)
HMD	**H**ead **M**ounted **D**isplay
HOCU	**H**ardened **O**perator **C**ontrol **U**nit
IDE	**I**ntegrated **D**evelopment **E**nvironment
IK	**I**nverse **K**inematics
KDL	Orocos **K**inematics and **D**ynamics **L**ibrary
OSL	**O**cean **S**ystems **L**ab
PCT	**P**ast **C**ut of **T**ime (over 5 minute 30 second time limit)
ROS	**R**obot **O**perating **S**ystem
ROV	**R**emotely **O**perated **V**echicle
VR	**V**irtual **R**eality

Chapter 1

Introduction

1.1 Background

Lichiardopol [2007] defines teleoperation as 'doing work at a distance'. It is the process by which an operator controls a robot remotely. The terminology used is vague as 'distance' may mean operating at a considerable physical distance, for example military robots exploring dangerous environments, or at a smaller scale, such as microscopic surgeries. The use of teleoperation has benefited a number of industries. These include space, military missions, mining, surgery and underwater explorations. In particular, teleoperation has proven to be very beneficial in many hazardous and unreachable areas, as well as areas that may be expensive for humans to access directly.

Deep-sea explorations were one of the first uses of teleoperation (Lichiardopol [2007]). A remotely operated vehicle (ROV) allows an operator to conduct a number of underwater tasks such as 'surveying, inspections, oceanography and different simple manipulation and work tasks' (Lichiardopol [2007]). Underwater manipulators, which are often attached to ROVs, are also controlled using teleoperation. The use of teleoperation allows an operator to control the manipulator's arm and end effector remotely from the surface.

Currently, there are many different styles of controllers used for teleoperation. For example, joysticks, teach pendants, radio remote controllers and game controllers (Zubrycki and Granosik [2015]). Without extensive training, these devices are often hard to operate well, as they often feel unnatural and unintuitive. As a result, the operator is likely to focus more on the controller rather than on the performance of the required task (Almeida et al. [2014]).

One solution that may lead to a more intuitive controller is the use of computer vision and gesture-controlled devices. Research into gesture control began in 1980 (Bhuiyan and Picking [2009]) and since then the technology and research have advanced dramatically. With the development of game controllers, such as the Wii and the Kinect, many people are now starting to have direct experience of these technologies. This research suggests that by providing the operator with a more intuitive control method, such as gesture control, operating the manipulator will become easier. As a result, it is likely that the operator will feel more confident using the technology, allowing them to focus more on the specified task.

1.2 Research Focus

Recently, there has been an increased interest in the use of hand tracking technology and natural user interfaces to allow users to interact with their environment (Taylor et al. [2016]). Hand tracking technology has been used predominantly in the games industry, with the creation of devices such as the Wii and Kinect. With recent advances in the accuracy and efficiency of these devices, it is now possible to use this technology within the field of robotics. This research aims to demonstrate that by using gesture-controlled technologies, the operator can become more precise with their movements when controlling a manipulator. In turn, this will allow the operator to control the manipulator in a more natural and intuitive manner compared to controlling it using a joystick-based controller.

1.3 Research Aims and Objectives

The key aims of this research are:

- *Explore* the different gesture controlled devices that are available
- *Decide* on a suitable gesture control device
- *Evaluate* different motion tracking options.
- *Develop* a controller using a more intuitive gesture controlled input
- *Compare* the gesture controlled controller to the current joystick controller

1.4 Research Relevance

Currently available teleoperation controllers often feel unnatural and unintuitive to use and require extensive training. Controllers that allow operators to use a more natural interaction to control underwater manipulators will save time and money. This is due to the reduced training required, enabling a larger number of operators to control underwater manipulators. With advances in the research and development of tracking software, virtual reality and haptics, robotics is now on the verge of delivering a more intuitive means of teleoperation for underwater manipulators. The findings of this research are expected to lead to future studies, where the advantages of an advanced teleoperation control system with visual and tactile feedback can be explored further.

Chapter 2

Literature Review

2.1 Introduction

The aim of this research is to find a more intuitive means of control to improve teleoperation of an underwater manipulator. This literature review aims to acheive the following:

- *Identify* different methods of control input
- *Critically evaluate* the best control input for teleoperation
- *Explore* different technologies available for the chosen control input
- *Explore* different motion tracking options

The findings of the literature review is organised as follow: Section 2 discusses different control input devices; alternative methods of gesture control are detailed in Section 3; and Section 4 describes different methods of motion tracking.

2.2 Control Input

2.2.1 Operator Control Unit

One of the most common methods of controlling a manipulator is the operator control unit (Zubrycki and Granosik [2015]; Bassily et al. [2014]). An operator control unit usually consists of a joystick and a computer. The joystick is used to control the position and velocity of the end effector, while the computer is used to calculate the necessary joint angles of the arm using inverse kinematics. This allows the manipulator to move the end effector to the desired location (Shim et al. [2010]).

The HDT Adroit-M Manipulator uses a Hardened Operator Control Unit (HOCU) (HDT Global [2016]). The joystick allows the operator to rest their hand comfortably whenever they need to take a break. However, it also has its disadvantages. The joystick often requires hand and arm movements that may be awkward and unintuitive. Additionally, it takes a skilled and experienced operator to use an operator control unit successfully (Shim et al. [2010]). However, it is often the case that skilled operators are not available (Zubrycki and Granosik [2015]). To address these issues, an alternative method of operation, which is more intuitive to use and does not require extensive training, is needed. Two possible alternative control methods are discussed below.

2.2.2 Voice Recognition

Voice recognition can act as an alternative approach to operator control units. In the study "A FRIEND for assisting handicapped people" (Martens et al. [2001]), speech recognition software, used by the semiautonomous robotic system 'FRIEND', allows spoken instructions to be compared to a set of approved words, which in turn carries out the required hardware movement. However, it does not give any scope for expanding these commands, enabling them to be used in more complex

environments. Additionally, for safety reasons, these spoken instructions are often long and complicated. Having complex sentence commands prevents background noise being confused as commands. However, this results in multiple sentences needing to be spoken in turn to carry out a string of tasks.

2.2.3 Gesture Control

In the paper "Gesture-controlled operator interfaces, what have we done and what's next", (Bhuiyan and Picking [2009]), the development of gesture-controlled technology over a period of 30 years is explored. Gesture controlled inputs are found to be particularly beneficial to operators that struggle with more conventional inputs i.e. mouse and keyboard.

Gestures are a primary form of communication for humans and are often found naturally in babies before they can speak. Bhuiyan and Picking [2009] define gestures as "non-verbal communication made with a part of the body". With recent advances in technological affordability, it is now possible to control electronic devices using these gestures as an input. Currently, there are many different methods of gesture sensing and, as a result, nearly all parts of the body can be sensed.

Gestures can be recognised using several different methods, such as accelerometers, wearables, gloves and cameras. These various technologies are often used in conjunction with one another which allows them to control the hardware needed to achieve the desired task. Cameras are now being used more than sensors as they are easier to use. This has resulted in more household products, such as laptops and TVs, having features that allow the possibility of gesture controlled interaction.

The majority of research that has been conducted to date within the gesture control field has been centred around hand gestures. This has mainly been achieved using data gloves which are connected up to a micro-controller. Additional focus has also been spent on head gesture recognition used alongside speech. However,

this has not been as predominant. Researchers are now moving away from the likes of glove sensing and moving towards image processing software.

The games industry is one of the main leaders in gesture control technologies. Consoles such as the Wii and Kinect both use gesture control. Industries such as simulation, training and education, as well as assistive learning also use gesture controlled technologies. It is expected that gesture control may become more mainstream due to the decrease in cost. These new technologies are also gradually becoming more intuitive and natural to use.

2.3 Gesture Controlled Technologies

2.3.1 Controller-based Gestures

Within gesture controlled technologies, there are a wide range of products that operate by direct contact with the operator. These include the Wii, tracking suits and exoskeletons as well as haptic controllers. Technologies such as the WiiMote are embedded with multiple sensors, such as accelerometers, that allow them to interpret the operator's gestures (Guna et al. [2014]). Another alternative method is the use of vision-based devices which often use markers attached to the operator's body. These markers allow the cameras to focus on specific areas of the operator's body (Du and Zhang [2014]; Kofman et al. [2005]).

By attaching the technology directly to the operator, the detection of movement is more reliable than using computer vision alone. As a result, the joint angle measurements are more precise (Breazeal and Scassellati [2002]). However, it does have its limitations. Technologies such as exoskeletons are cumbersome and difficult to transport (Goncalves et al. [1995]). Additionally, having markers attached to the operator's body may hinder and limit the operator's movements (Du and Zhang [2014]).

2.3.2 Wired Gloves

Wired gloves allow the operator's finger joint angles to be accurately measured. This allows the operator to control the end effector gripper in an intuitive manner and allows the gripper to mimic the operator's hand gestures. The use of the glove is found to be particularly effective when the gripper is of similar structure to a human hand (Zubrycki and Granosik [2015]). As discussed in Zubrycki and Granosik [2014], due to the glove being directly attached to the operator's hand, there is no effect from external environmental conditions, such as light, that may affect the measurements.

Wired gloves are mechanical and electrical and as a result, they come with added issues such as wearing out. They need to be regularly calibrated as the resistance of the sensors can vary greatly. Additionally, as the operator's hand is in the glove, their hand can not be used to perform any other tasks (Zubrycki and Granosik [2014]) and limits the potential for natural interaction (Taylor et al. [2016]).

2.3.3 Single Camera

Many gesture controlled technologies are now using cameras and computer vision. Single cameras or monocular cameras can be found on most mobile phones. As examined in Goncalves et al. [1995], it is possible to use an estimated image, compared to the visual image, to get a sense of depth. This allows gestures to be recognised without the use of body markers, allowing the operator to have a wider range of movements.

A disadvantage of this method is the fact that depth has to be relative. A single camera is unable to gauge the depth of an isolated point. This results in a low level of accuracy.

2.3.4 Depth-aware Camera

The Kinect is a great example of a depth-aware camera. It contains two cameras and a receiver: one infrared camera transmitter and receiver, which are used for depth detection and a standard camera which can be used for visual recognition (Du and Zhang [2014]). It can detect full body gestures (Guna et al. [2014]) and, as stated in Afthoni et al. [2013], it can detect over 15 joints. These include the head, neck, torso, shoulders, elbows, hands, knees, hips and feet.

The Kinect is fairly accurate when detecting the operator's joints, with errors around a few millimetres. It can also recognise a number of set gestures which can be further added to by the operator (Microsoft [2016]). Due to the distance being detected with infrared sensors, the Kinect is not affected by changes in environment lighting (Du and Zhang [2014]). As a result, it is fairly robust to changes in position and location (Taylor et al. [2016]). The Kinect allows the operator to connect with the manipulator in a more natural and intuitive way. The use of vision only sensing removes the need for markers, sensors and cables that may hinder the operator's movement (Du and Zhang [2014]).

As stated on the Microsoft website (Microsoft [2016]), the Kinect has a sensor range between 0.8 - 4 meters. This large working volume decreases the chances of the operator's body or hands going out of range (Taylor et al. [2016]). This distance can be decreased using the Kinect's "near mode" which shortens the range to 0.5 - 3 meters. Its depth accuracy has a standard deviation of around 1.5 cm (Bassily et al. [2014]). As a result of its limited depth accuracy, while the Kinect is good at detecting the arm and body, its accuracy of finger tracking is very limited Guna et al. [2014]; Taylor et al. [2016]).

2.3.5 Stereo Camera

Two good examples of gesture controlled devices using stereo cameras are the Leap Motion and 3 Gears (Bassily et al. [2014]). The Leap Motion has sub-millimetre

accuracy and can track all ten fingers at once (Bassily et al. [2014]). Both the Leap Motion and 3 Gears are specially designed for hand gesture recognition (Zubrycki and Granosik [2014]) and provide position, orientation and joint angles for every visible finger and hand (Zubrycki and Granosik [2014]). These devices produce a limited amount of data, but the hand and finger data they produce is a lot more accurate than the data provided by other technologies, such as the Kinect (Marin et al. [2014]).

The field of view for the Leap Motion controller is an inverted pyramid shape, with the point at the sensor's centre (Guna et al. [2014]). As the operator's hand is moved further away from the sensor, the accuracy drops (Guna et al. [2014]). It is stated on the Leap Motion developer website (Leap Motion [2016*d*]) that the sensor range is between 25-600mm (0.025 - 0.6 meters).

As with all technologies however, it has its disadvantages. Stereo cameras often have an issue with instability and tracking each hand when the operator's hands are placed on top of each other. The Leap Motion is also unable to detect angles when hands are smooth i.e. in gloves, and finger tracking is not as accurate when the angles between the fingers are large. In addition, constant light conditions are needed (Zubrycki and Granosik [2014]).

Leap Motion, however, has recently developed a new software called Orion (Leap Motion [2016*e*]) which is stated to have the following improvement:

- Enhanced tracking capabilities
- Robust to complex environments
- Expanded range
- Vastly improved grab and pinch
- Faster hand recognition

The Orion software containing these improvements is currently only available for the Windows operating system.

2.4 Motion Tracking

2.4.1 Motion Primitives

There are a number of ways to drive a manipulator once the gestured input has been obtained. Motion primitives involve saving a set of pre-programmed motions (Shim et al. [2010]). The setting of these motion primitives can either be performed by manually moving the arm and saving the joint angles or by manually inputting the required joint angles (Martens et al. [2001]).

Some technologies such as the Leap Motion and the 3 Gears already come with a number of preset recognised gestures (Zubrycki and Granosik [2014]). The Leap Motion also allows the operator to set a number of custom gestures (Bassily et al. [2014]). These can then be categorised using the cameras on these gesture controlled devices. This can then be used to give the operator feedback on the speed, direction, position and orientation of the gesture performed (Zubrycki and Granosik [2015]).

Motion primitives are useful when simple commands are required. However, when trying to achieve more complex commands or a string of flexible commands, motion primitives can prove to be limiting (Du and Zhang [2014]). Additionally, with a fixed set of allowed gestures it forces the operator to remember all the possible gesture commands. This may prove difficult to do under stressful circumstances.

2.4.2 End effector Tracking

Compared to motion primitives, end effector tracking allows the operator to interact with the manipulator in a more intuitive manner. End effector tracking relies on the operator's hand being tracked in free space. Inverse kinematics is then used in order to work out the required joint angles for the arm in order to move the end effector to the specified location (Shim et al. [2010]; Du and Zhang

[2014]). End effector tracking allows the operator to concentrate on the necessary task without having to remember a set of allowed gestures (Du and Zhang [2014]). Hand gesture tracking devices such as the Leap Motion, as well as wrist markers, allow hand and finger data to be accurately measured, which can then be used for end effector tracking. However, due to the fact that only the position of the end effector is controlled, the operator has no control over the movement of the arm joints. This lack of control of the arm may become an issue if the requested position is outside the manipulator's workspace (Shim et al. [2010]).

2.4.3 Motion Retargeting

One of the more complex methods of controlling a manipulator is motion retargeting. This form of tracking involves mimicking the skeleton of an operator i.e. hip, shoulder, arm, fingers and getting the joint angles between them. The joint angles can then be mapped directly onto the manipulator (Breazeal and Scassellati [2002]; Zubrycki and Granosik [2015]). Compared to end effector tracking, motion retargeting requires a method of obtaining all the required joint angles. The joint angles and locations will vary from person to person due to varying height and body sizes. As a result, additional calibration may have to be implemented (Du and Zhang [2014]).

2.5 Conclusion

From reviewing the literature, it is clear that at present several problems are faced when using an operator control unit. Despite this, it is suggested that gesture control may be a possible solution to improve controller intuitiveness and ease of use. It is evident that there are a number of different gesture controlled devices, each with their own methods of extracting data. To have a natural input device, the operator's manoeuvrability should not be limited. Although thinner data gloves are being developed which would allow for better manoeuvrability, they are

often more expensive than camera-based devices. Data gloves are also very fragile which results in the hardware frequently breaking.

As explained, single cameras are not very accurate and therefore would not be a good choice as a teleoperation input. Depth-aware cameras, such as the Kinect, and stereo cameras, such as the Leap Motion, each have their advantages and disadvantages. One of the issues with the Kinect is its calibration procedure. For the Kinect to find the operator's position, the 'psi pose' (Fig. 2.1) must be performed. This 'psi pose' position has to be held for an extended period and has to be repeated every time the Kinect program is restarted. This can result in the operator's arms being fatigued quickly. Additionally, the Kinect device is currently inadequate at detecting hand orientation and fingers, although work is being done to improve this. While the manipulator arm would be easily controlled using the Kinect, it would be hard to control the end effector's fingers without the aid of an additional device. The Leap Motion, on the other hand, has very precise hand and finger detection. As a result, for the purposes of this research, the Leap Motion device was chosen to control the manipulator.

FIGURE 2.1: Psi pose (creativec0d3rl [2016])

The control method is constrained since the Leap Motion does not detect the operator's whole arm. This rules out the possibility of using motion retargeting. While motion primitives would be a suitable option, it limits the amount of control

the operator has over the manipulator. This is due to it only being able to perform a set of fixed movements depending on the gesture recognised. The most promising control option is end effector tracking. The Leap Motion detects the operator's hand in 3-dimensional space. This allows the required end effector position to be set. Inverse kinematics is then used to get the required joint states needed to position the end effector in the correct location.

Chapter 3

Research Methods

3.1 Introduction

Many underwater vehicles are equipped with manipulators which allow them to carry out tasks such as drilling or inspections (Shim et al. [2010]). One example of an underwater manipulator is the HDT Adroit-M Undersea Manipulator. This manipulator has an end effector that consists of two fingers and a thumb. Each finger has one degree of freedom and the thumb has two degrees of freedom. The HDT Adroit-M Undersea Manipulator is currently teleoperated using a Hardened Operator Control Unit (HOCU) (HDT Global [2016]). While the HOCU controller allows the operator to teleoperate the manipulator, it requires an experienced and skilled operator to control it. This is due to it being unintuitive, making it hard for the operator to know what movements need to be performed with the joystick in order to achieve the desired task.

This paper presents an alternative method of control using gesture-recognition technology. The premise being, that the use of a gesture controlled input device should create a more intuitive controller and improve ease of use. The research behind this paper was conducted to see if a Leap Motion device could be a possible means of controlling an underwater manipulator and whether it outperforms the currently available HOCU device. The desired outcomes are as follows:

- *Run* a working simulation using the Leap Motion to control the motion of the underwater manipulator using end-effector tracking
- *Control* the physical HDT Adroit-M Undersea Manipulator by mimicking human hand gestures captured by the Leap Motion sensor.
- *Contrast* the performance of the Leap Motion controller to the HOCU controller

This chapter covers the research strategies applied, the method of data collection and analysis, as well as the potential problems and limitations.

3.2 Research Strategy

An experimental research strategy was used to test the hypothesis that 'a gesture control input would provide a more intuitive method of controlling the HDT Adroit Underwater Manipulator compared to the current HOCU controller'. This strategy was tested using timed trials as well as a questionnaire.

3.2.1 Sending and Extracting Data

Data needs to be extracted from the Leap Motion, analysed and then sent to the manipulator to control it. The steps needed to complete this process are explained below.

3.2.1.1 ROS

The software within the HDT Adroit Manipulator uses Robot Operating System (ROS) drivers. ROS is an open-source operating system made specifically for robotic software development. As the design and production of robots have increased there has become a need for a common operating system. This allows software to be standardised and allows for better collaboration between developers. The main programming languages for ROS are Python, C++ and LISP. The

majority of the code written for this project is in C++. However, the code used to extract the Leap Motion data is written in Python, since an existing open source library is used. Within ROS there are a number of key libraries:

roscore: roscore is needed before starting any ROS applications. roscore allows ROS nodes to communicate with each other. This allows messages to be passed between classes such as in a publish-subscribe methodology as described in section 3.1.

rosbag: rosbag allows ROS to record and playback ROS topics i.e. messages. This allows the output signals to be analysed.

rqt_plot: rqt_plot provides a way of graphically plotting ROS topic signals. This allows position and orientation of topics to be analysed as well as changes in joint state values.

RViz: RViz is a three-dimensional visualisation tool that is used to simulate robots using ROS. It is often necessary to simulate robot actions in a simulator before testing it on the real robot. By testing the controller first within the simulator it minimises the chances of damaging the actual robot and allows for many iterations of the program to be tested in succession.

Transform: ROS transforms allow the operator to view different coordinate frames in the simulation relative to a specified world frame.

Marker: Feedback messages are displayed on the RViz simulation which provide the operator with extra information. The messages are displayed as ROS text markers. The following message are displayed to the operator:

- Place your right hand over the Leap Motion and press the ‘s’ key to start
- Position not reachable

- Program is paused
- Program is locked, place your right hand over the sphere to unlock

3.2.1.2 Publisher and Subscriber

One of the main forms of passing data within ROS is using the publish-subscribe model. For this project the main publish-subscribe paths are shown in Figure 3.1.

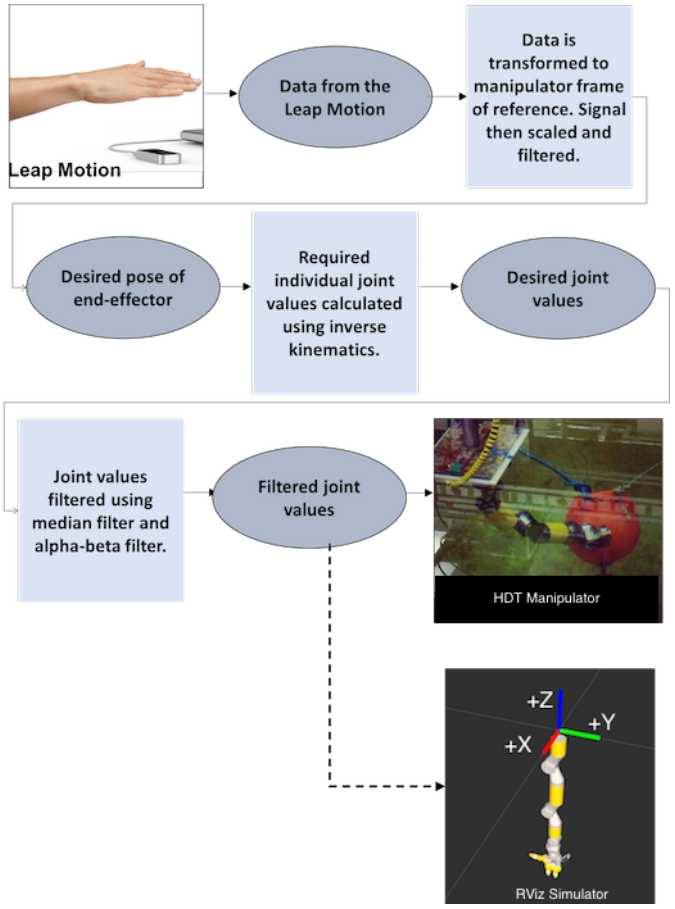


FIGURE 3.1: Main data workflow

The publish-subscribe model allows for easy scalability and flexibility. Data that is published can be accessed by multiple subscribers which allows the same data to be used for multiple purposes. For example, the joint state data is used to update the HDT manipulator as well as updating the simulated manipulator

within RViz. When data is published it is given a unique name. This unique name is then used in order to subscribe to the required data.

When creating a subscriber a spinner must be used. When data is published it is added to a queue. A spinner continuously checks this queue to see if any new data has been published that needs to be passed on to the relevant subscribers. Without a spinner any new data will be ignored. ROS has two types of spinners; single-thread spinner and multi-thread spinner.

- Single thread spinner: This is the default spinner. Data on the queue is processed one at a time, blocking the sending of any other data to other subscribers in the mean time.
- Multi-threaded spinner: This allows data to be sent in parallel. The number of parallel threads are specified when initialising this spinner.

When creating the publisher it is important to create it during the initialisation of the class. If the publisher is created and a message is published immediately after, the publisher will not have sufficient time to be fully initialised. This will result in the data failing to be sent.

3.2.1.3 Extracting Leap Values

As previously discussed, ROS is an open source platform with many contributors. This shared software removes the need to rewrite code from scratch every time. A Python based *leap_motion* library already exists in ROS for extracting hand and finger information from the Leap Motion (ROS [2016*d*]). The *leap_motion* package contains two classes: sender.py and skeleton_send.py. The sender.py class publishes LeapROS messages, while the skeleton_send.py publishes the relative finger joint positions and orientations from the Leap Motion as a transform. Both of these features are required and as a result, both of these classes were merged to create a custom class. This allowed for one class to be run that can achieve all the desired tasks rather than having to run two separate classes. This custom

class detects when the operator's right hand is over the Leap Motion and sends out data in the form of a LeapROS message. It also publishes the Leap Motion skeleton transforms. These LeapROS messages include:

- **Single hand 3D Palm Position:** This data is used to get the current position of the operator's hand relative to the Leap Motion.
- **Single hand 3D Palm Normal Vector:** not used.
- **Single hand 3D Hand Direction Vector:** not used.
- **Single hand 3D Palm Pose (pitch, yaw, and roll):** This data gives the relative orientation of the operator's hand.
- **All Finger joints ['thumb', 'index', 'middle', 'ring', 'pinky'] positions:** This data is used to get the relative positions of the finger joints.
- **Raw camera images:** not used.

This Leap Motion data is then read to control both the position and orientation of the end-effector as well as the grip of the fingers.

There were also a number of changes made to the leap_interface.py class. The roll, pitch and yaw values were converted from degrees to radians. This change was made in order to keep all the values as standard SI units. Additionally, the method to calculate the roll, pitch and yaw was altered. In the original leap_interface.py class, the yaw is calculated relative to the palm normal and roll relative to the hand direction. However, from the Leap Motion website, Leap Motion [2016*b*], it states that pitch and yaw should be calculated using the direction vector and the roll should be calculated using the normal vector.

$$pitch = hand.direction().pitch(); \tag{3.1}$$

$$yaw = hand.direction().yaw(); \tag{3.2}$$

$$roll = hand.palmNormal().roll(); \tag{3.3}$$

Once the Leap Motion information is received from the LeapROS message, the data has to be transform, scaled, filtered and the inverse kinematics has to be calculated, before a jointState message can be published. Once the jointState values are received, the manipulator's ROS drivers can then update the manipulator's position.

3.2.2 Executing the Controller

Before the operator can teleoperate the HDT manipulator, a number of programs have to be executed.

- roscore : Allows ROS nodes to communicate with each other.
- RViz simulator: Simulates the manipulator's actions and displays feedback messages to the operator.
- leap_interface: Sends out data in the form of a LeapROS message and publishes the Leap Motion skeleton transforms.
- keyboard: This is used for both starting and pausing the controller
- controller: Provides the conversion from Leap Motion position to the manipulator end-effector position.

3.2.3 Population

Ideally, a large random sample population would have been used. However, due to health and safety restrictions in the Ocean Systems Lab, the target population was chosen using convenience sampling. Although this sampling method does not allow for generalised results, it provides an exploratory comparison between the two controllers. While the number of members in the Ocean Systems Lab is small,

the sample size was limited further due to a lack of volunteers. The final population consisted of three male subjects. Further trials would need to be conducted using a larger and varied population sample to obtain more representative results. The following selection criteria was used:

- Participants who have never used the HOCU or Leap Motion before. No previous experience with either controller was desired to avoid bias.
- Limited to participants within the Ocean Systems Lab due to health and safety reasons

3.2.4 Task

The task was set up to compare not only the performance of the devices but also the manoeuvrability, comfort and ease of use. The task was to touch two buoys in the fastest possible time. There was a total of 3 trials on each device. Any trials taking longer than 5 minutes 30 seconds were viewed as a failed attempt.

3.3 Data collection

This section discusses how the data was collected including the procedure, setup and metrics.

3.3.1 Procedure

The participants were asked to complete the task three times, in succession, on each device. To prevent candidates from unintentionally causing a biased result, by comparing the controllers against each other, a questionnaire was asked immediately after the three time trials on each device were completed. The bias was further minimised by having different candidates start on different devices.

3.3.2 Setup

The manipulator was placed in the tank with two buoys fixed either side of it (Fig. 3.2, Fig. 3.3).

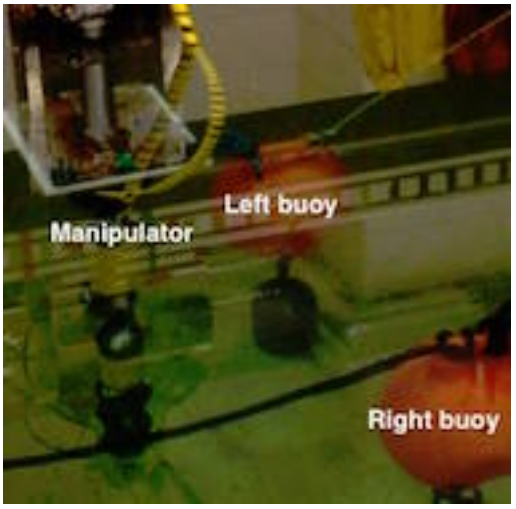


FIGURE 3.2: Manipulator and buoy setup

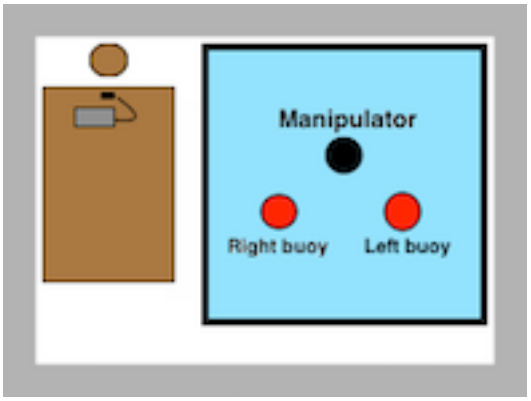


FIGURE 3.3: Manipulator and buoy setup

The operator was sat at the desk and positioned so that the simulated manipulator and HDT manipulator were at the same orientation (Fig. 3.3). Ideally, a wide lens camera would have been attached to the manipulator to allow the operator to view the manipulator and its surroundings on the computer screen. This, unfortunately, was not possible due to a number of complications. The only way to mount a camera on to the manipulator is by using cable ties attached to

the base. When the camera is loosely attached with the cable ties, the camera fails to stay in a fixed position when the manipulator moves. When the cable ties are fastened tightly, the only view is of the base of the manipulator. As a result, it was found that viewing the manipulator directly was more beneficial.

3.3.3 Metrics

Several aspects were focused on when comparing the devices:

- **Time taken to hit each of the buoys:** This demonstrates how easy it is to complete a specified task using each controller.
- **Intuitiveness:** The aim is to have a controller that operators find natural to use.
- **Ease of taking a break:** It is important that the operator can pause the system and walk away from it.
- **How painful or tiring it is to operate the device:** The controller should not be painful to use.
- **Precision:** The controller should be precise in order to pick up specific objects.
- **Controllability:** The operator should feel as though they are in control of the controller at all times.

3.4 Data Analysis Framework

There were three participants and each had three trials on both the Leap Motion and HOCU devices. For each trial, the following were recorded:

- Elapsed time to touch the left buoy
- Elapsed time to complete the full task

For each of the above measurements, on each device, the average across all participants' trials were calculated. The average of each of the devices was then compared. The questionnaire consisted of five questions, each with a rating between 1 (low) and 5 (high). The questionnaire was used to assimilate how the participants felt about each device. The questionnaire values were averaged per question across the three participants for each device. These averages were then compared between the two devices for each question.

3.5 Limitations and Potential Problems

While the program was completed successfully, there were a number of limitations and potential problems that had to be overcome.

3.5.1 Software Limitations

The physical HDT arm is controlled by a 'JointState' message. This jointState message contains the following parameters:

- position
- velocity
- effort

In this program both the position and the velocity of the joint states are varied. The RViz simulation allows the program to be tested in simulation before applying it on the physical HDT manipulator. It is, however, limited when it comes to controlling the joint states. While the simulator has no problem in updating the position of the manipulator, it fails to update the velocity. This results in the velocity of the manipulator within the simulator remaining constant, regardless of the actual value set. Consequently, precautions had to be taken when testing the control on the actual arm with varying velocities.

3.5.2 Hardware Limitations

Communication is one of the biggest control problems for teleoperated systems (Lichiardopol [2007]). When using the hardware, the connection was a major issue. The drivers were often disconnected which resulted in the manipulator being unresponsive until it was restarted. It is thought that the issue was due to the connector having a loose pin that became disconnected when the manipulator moved erratically. As a temporary fix, the cable was attached to the manipulator using cable ties to keep the cable at the correct angle.

As well as the manipulator having connection issues, the end effector also suffered hardware faults. The thumb and fingers have a total of four joints, however only one of these joints was working. As a result, the end effector was removed completely from the manipulator. This minimised the type of tasks that could be completed as there was no way of gripping objects. Unfortunately, due to this, the control of the end-effector fingers was only tested in the simulation.

Finally, the camera proved to be a limitation. The camera that was available was only able to stream to an external monitor rather than the laptop. The camera had to be mounted on with cable ties and the only suitable location to attach these are at the base of the manipulator. At this position however, the operator could not see much of the manipulator nor its environment, as it was hard to keep the camera fixed at the required position. As a result, the camera was not used and the operator had to look at the manipulator directly.

3.5.3 Associated Risks

While working on this project there were a number of associated risks that had to be mitigated. The risks and their mitigation plans are detailed below:

- **Tripping over cables:** Make sure cables are put out of the way or covered with anti-trip tape.

- **Falling in the water and drowning:** Make sure there is someone else in the room when working near the water tank.
- **Electric shock if water comes into contact with faulty electrical equipment:** Keep electrical cables away from the water.
- **Injuring yourself from being hit by the manipulator if within reach and travelling at high speeds:** Make sure no-one is within reaching distance of the arm when powered on.
- **Injuring yourself when trying to lift weighted buoys out of the water:** Ask for help if the buoys are too heavy to manage yourself.

Chapter 4

Technical Challenges

During the development of the controller, there were a number of challenges that needed to be addressed.

4.1 Coordinate Frames

When controlling the HDT manipulator, it is important that both the manipulator and the Leap Motion have the same frame of reference. As the operator moves their hand from left to right across the Leap Motion, it is expected that the manipulator moves in the same manner, mimicking the operator's movements. However, as can be seen from the figures below (Fig. 4.1, Fig. 4.2) the Leap Motion and manipulator have different coordinate frames. To correct this, the following conversions are made when reading in the Leap Motion data.

Leap Motion X-axis = manipulator Y-axis
Leap Motion Y-axis = manipulator Z-axis
Leap Motion Z-axis = manipulator X-axis

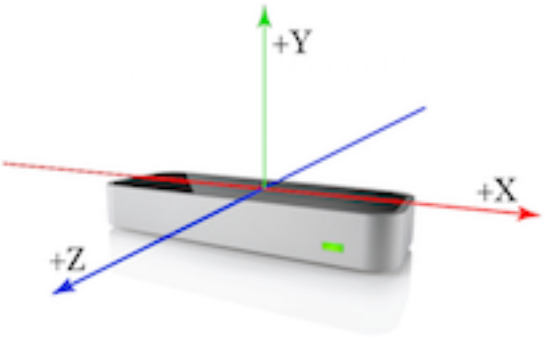


FIGURE 4.1: Leap Motion coordinate system (Leap Motion [2016*a*])

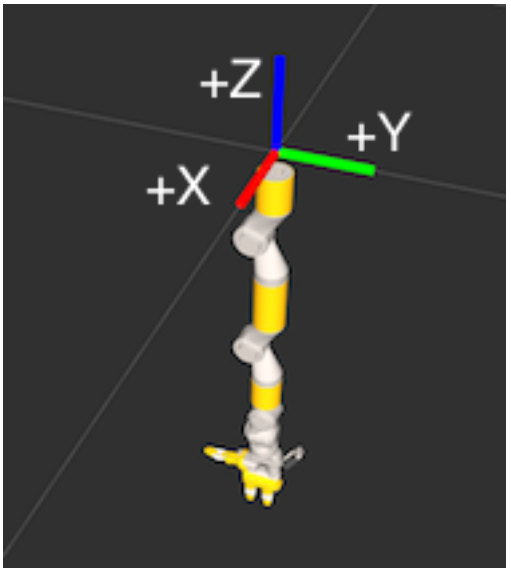


FIGURE 4.2: HDT Adroit Manipulator coordinate system

The orientation of the Leap Motion i.e. whether the front light is pointing towards or away from the operator, can partly be ignored. This is due to the fact that the Leap Motion has an automatic orientation feature. This feature results in the +z-axis always being at the side of the Leap Motion that the operator is on. However, this feature only works if the Leap Motion is placed flat on a surface with the y-axis pointing straight up (Leap Motion [2016*a*]).

4.2 Calibration

4.2.1 Default Configuration

When the HDT manipulator drivers start up, the manipulator resets back to its default configuration. This default configuration consists of all the joint state values being set to zero, forcing the manipulator to adopt a straight down position (Fig. 4.2). Both the position and orientation of the end effector are set relative to the manipulator's base link which is located at the top of the arm (Fig. 4.5).

4.2.2 Initial Configuration

Due to the manipulator resembling a right hand, this controller is programmed with the intention of the operator controlling the manipulator using their right hand. Originally, the initial configuration (Fig. 4.3) was achieved by setting joint 7 of the manipulator to be equal to -1.55. This is the minimum position the joint 7 value can reach without hitting its joint limit value of -1.57. When the operator initially starts with their hand over the Leap Motion their wrist is bent. It was thought that by having the manipulator's initial position with the wrist also bent it would provide a more intuitive controller. The trouble with this initial configuration is that the manipulator is already at its full extension. However, when the operator initially places their hand over the Leap Motion they are not at their full extension. This often resulted in the operator lowering their hand, only to hit a joint limit.

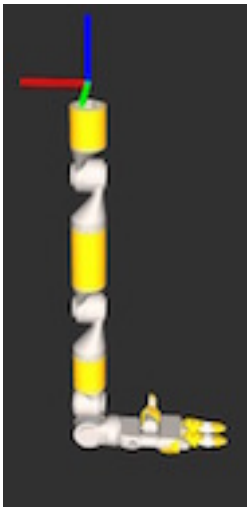


FIGURE 4.3: Old initial configuration

As a result of this limitation, an alternative initial configuration was chosen. When the operator places their hand over the Leap Motion, as well as having a bent wrist, they also have a bent elbow, with their hand perpendicular to their shoulder (Fig 4.4). To ensure that the control of the manipulator is as intuitive

and natural as possible, both the operator's arm and the manipulator (Fig. 4.5) need to have roughly the same starting pose.

FIGURE 4.4: Operator initial configuration

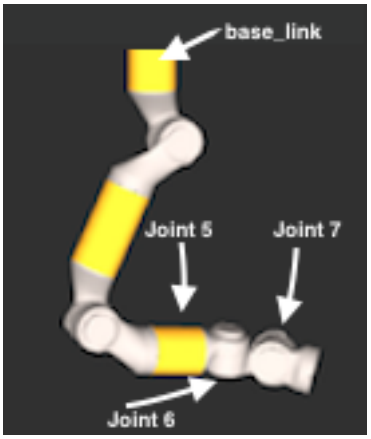


FIGURE 4.5: HDT initial configuration

4.2.2.1 Position

The operator's hand was used to position the end effector at the desired position. The manipulator was then paused and the relative position and orientation were extracted relative to the base link. The resulting end-effector position is set as follows:

$$posX = -0.268 \tag{4.1}$$

$$posY = -0.070 \tag{4.2}$$

$$posZ = -0.614 \tag{4.3}$$

4.2.2.2 Orientation

As well as the position of the end-effector, the initial orientation of the end-effector has to also also be set. The values were also extracted from the relative position

between the base link transform and the end effector transform within the RViz simulator.

$$roll = 3.136 \tag{4.4}$$

$$pitch = -1.343 \tag{4.5}$$

$$yaw = -3.134 \tag{4.6}$$

Once these values are set, a transform is then published with the required position and orientation. Inverse kinematics is then used to calculated the required joint angles needed to place the end effector in the required position with the requested orientation. Then joint values are then sent to the manipulator to set the initial configuration of the manipulator.

4.2.3 Desired Configuration

The pose of the hand above the Leap Motion is taken to be relative to the last read position and orientation. Once the initial position is set, a message appears in the RViz simulator telling the operator to "place your right hand over Leap Motion and press 's' to start". The operator has to hold their hand steady above the Leap Motion device as this allows a baseline position and orientation to be set. The desired position and orientation values can then be calculated relative to this baseline.

4.2.3.1 Position

Due to the Leap Motion and the manipulator having different coordinate frames, the change of hand position in the x axis is calculated using the following equation:

$$changeInXAxis = oldLeapPosZ - currentLeapPosZ \tag{4.7}$$

To calculate the new manipulator position, the change in Leap Motion position must first be scaled. The scaling factor defines how sensitive the change in position is relative to the movement of the operator's hand. The scale factor was chosen to be 0.005. This scaling factor was chosen by experimenting with different values and analysing which value allowed the system to behave in the most natural manner.

$$scaledChangeInXAxis = (changeInXAxis * scale) \tag{4.8}$$

Once the scaled value is calculated, the required HDT manipulator position in the x-axis is calculated using the following equation:

$$HDTPosX = HDTPosX - scaledChangeInXAxis \tag{4.9}$$

This process is repeated for both the y and z axis.

4.2.3.2 Orientation

By changing the manipulator from its default straight down configuration into this initial bent configuration, it affects the orientation control of the manipulator. The orientation is controlled by the three end effector joints which are labelled joint 5, joint 6 and joint 7. When the manipulator is straight down, the joints control the orientation as follows:

Joint 7 = pitch Joint 6 = yaw Joint 5 = roll

However, when the manipulator is in its bent configuration, the joint control for the orientation is changed:

Joint 7 = pitch Joint 6 = roll Joint 5 = yaw

This is due to a change in joint 7's coordinate frame between the initial and default configuration. As a result, the Leap Motion's hand orientation is set relative to the the initial orientation described in equations 4.4, 4.5 and 4.6. The yaw and roll are switched due to this change in configuration.

$$roll = -1 * LeapYaw \tag{4.10}$$

$$pitch = LeapPitch \tag{4.11}$$

$$yaw = -1 * LeapRoll \tag{4.12}$$

A ROS coordinate transform is then created and published with the desired position and orientation of the end effector relative to the base link. This transform is visible within the RViz simulator and allows the operator to see where their hand is relative to the manipulator. This information allows the operator position their hand accordingly and make any corrections needed.

4.3 Controlling the Fingers and Thumb

The fingers and thumb on the end effector also need to be controlled. By controlling these joints, it allows the operator to grasp different objects. The relative finger joint positions and orientations are sent from the Leap Motion (Fig. 4.6).

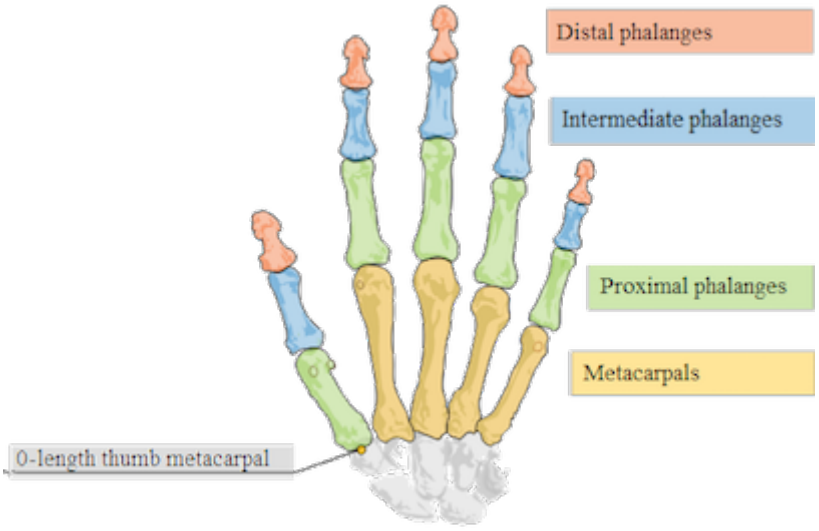


FIGURE 4.6: Bones detected by Leap Motion

Each of these fingers and their corresponding joint positions are published as a transform which are visible in RViz. The finger grip is calculated using the relative position and orientation between two bones.

4.3.1 Thumb

The end effector thumb has two degrees of freedom which control the thumb base and the thumb proximal joints. The thumb base controls the left to right movement while the thumb proximal controls the forward and backward movement. These two grip positions are calculated using the relative rotation between the tip of the thumb and index finger metacarpal. The thumb base and thumb proximal joint values are calculated as follows:

$$thumb\_base = (x\_axis\_rotation * baseScale) + baseCorrection \tag{4.13}$$

$$thumb\_prox = (y\_axis\_rotation * proxScale) + proxCorrection \tag{4.14}$$

where:

baseScale = 2

baseCorrection = 0.5

proxScale = 1.5

proxCorrection: =1.2

The scale and correction parameter values were calculated experimentally. These parameters are needed in order to control the end effector's thumb from the Leap Motion data. The scale is used to control the speed at which the end effector's thumb travels relative to the operator's thumb. The aim is to have them matched so when the operator moves their thumb the end effector mimics it. The correction

value is used to set the end effector's thumb's initial position. The end effector should reflect the operator's thumb when it is in the neutral position. This allows for the operator's and end effector's thumb position to remain synchronised.

4.3.2 Index and Middle Finger

The end effector has two fingers which each have one degree of freedom controlling the finger proximal joint. The grip values of the index and middle fingers are calculated using the relative orientation between each fingertip and the corresponding metacarpal. The rotation between the two bones are calculated and the roll, pitch and yaw are then extracted in radians. The roll is converted into degrees and is then scaled. No calibration was needed when setting the finger's as the operator's fingers naturally are close to being parallel with the palm, which is the same as the end effector default position. The following equations are used to get the finger proximal values.

$$roll\_in\_degrees = abs(roll * 180.0/\pi) \tag{4.15}$$

$$finger\_proximal\_vaue = roll\_in\_degrees * 0.01 \tag{4.16}$$

4.4 Inverse Kinematics

Once the end-effector position and orientation are specified, inverse kinematics is used to calculate the corresponding joint values needed to move the end effector to the given position. Inverse kinematics is complex as it can often result in multiple solutions. There are two main approaches taken when solving inverse kinematics; analytical and numerical. The analytical approach is mainly used when there are a small number of links. The joint values are solved using geometry or algebraic equations. However, it is often the case that a number of solutions may be found, in which case additional constraints have to be applied. The numerical approach,

also known as the iterative method, is more general and is used to find solutions for more complex chains. Each of the joints are analysed in turn, and the rotation needed for the end effector to move towards the required position is calculated. pr2_moveit_tutorials (ROS [2016*c*]) contain a good tutorial on how to use the RobotModel and RobotState classes to calculate the inverse kinematics.

4.4.1 Orocos Kinematics and Dynamics Library

Within ROS, the most common inverse kinematic algorithm plugin is the Orocos Kinematics and Dynamics Library (KDL) (Beeson and Ames [2015]). KDL uses a numerical inverse kinematics implementation. The KDL solver has a number of issues. These include "Frequent convergence failures for robots with joint limits; no actions are taken when the search becomes 'stuck' in local minima; inadequate support for Cartesian pose tolerances; no utilization of tolerances in the IK solver itself" (Beeson and Ames [2015]).

As a result of these issues, the KDL solver often returns false-negative results (Beeson and Ames [2015]). False-negative results imply that the solver fails to find a solution when a solution exists.

4.4.2 TRAC-IK

Recently, a new inverse kinematics plugin has been released called trac-IK, which is more reliable at finding solutions and finds solutions quicker. Trac-IK runs two solvers simultaneously, once one of the solvers finds a solution, the solver stops.

When using the trac-IK kinematics plugin there are a number of parameters that can be set (track IK [2016*b*]):

- 'kinematics_solver_timeout': Specifies how many seconds it takes to run the solver before the function times out.

- 'position_only_ik': Specifies whether to only use the position alone to calculate the inverse kinematics or whether to use both the position and orientation.

- 'solve_type': Calculates the inverse kinematics using different equations. The allowed solve types are Speed, Distance, Manipulation1 and Manipulation2. The default solve_type is Speed.

For the purposes of this study, it was chosen to keep the 'kinematics_solver_timeout' parameter fixed at 0.005 seconds. However, different tests were carried out in order to choose the boolean value for the 'position_only_ik' parameter as well as the ideal solve type.

When using the kinematic plugin, the majority of the parameters are defined within a file named kinematics.yaml. However, during the testing it was apparent that setting the 'solve_type' and 'position_only_ik' parameters in this file had no effect and the default values were always being used. To overcome this issue, the value of these two parameters were set directly in the controller program. When initialising any ROS program, a node is created. A ROS nodeHandle was then created for this given node. Using the setParam method of this nodeHandle package, both the 'solve_type' and 'position_only_ik' parameters were then set.

4.4.3 Position only Inverse Kinematics Parameter

The manipulator end effector has a position and orientation. However, when calculating the inverse kinematics, the requested orientation of the end effector is optional. When the boolean parameter 'position_only_ik' is set to true, the orientation is ignored. Instead, the orientation of the end effector is allowed to take any value that allows the end effector to reach the required position.

The graphs below investigate the behaviour of the end effector using position only compared to using both position and orientation. For each case, the resulting position and orientation are shown for the end effector y-axis. The solve_type

parameter is fixed as the default solve type. The requested value is shown in blue and actual value of the end effector is shown in red. The x-axis represents the joint state value while the y-axis is time.

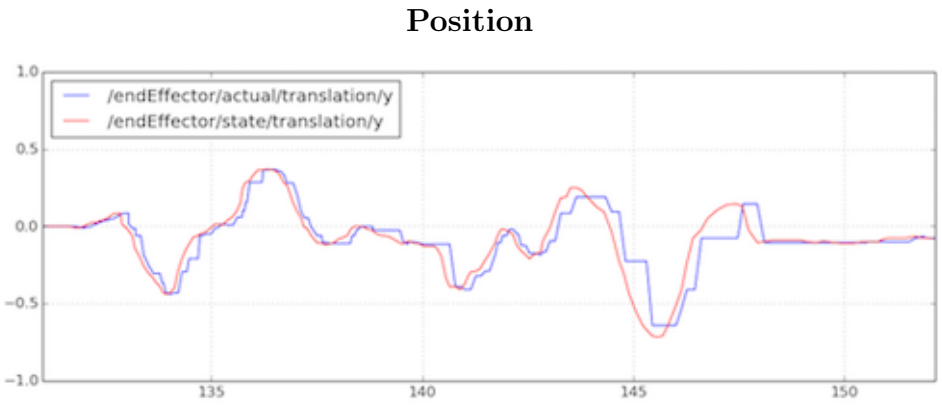


FIGURE 4.7: Position of end-effector y-axis with position_only parameter set to true

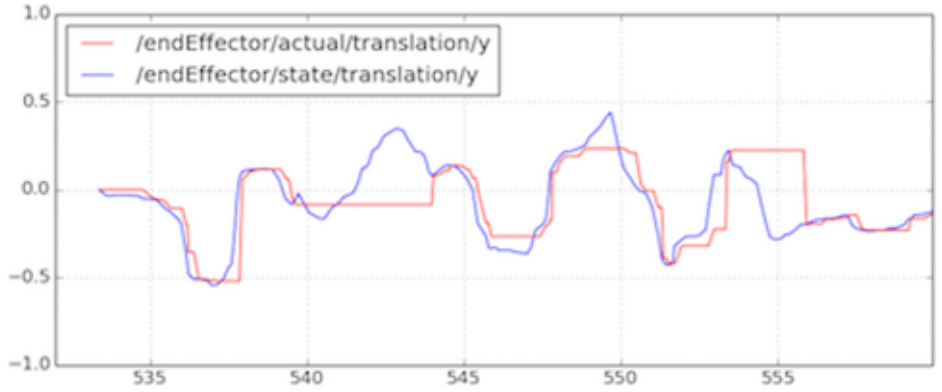


FIGURE 4.8: Position of end-effector y-axis with position_only parameter set to false

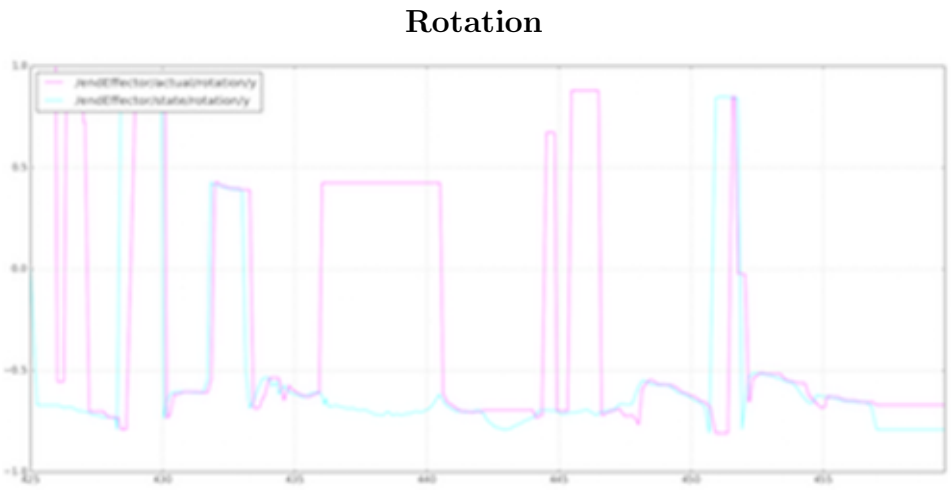


FIGURE 4.9: Rotation of end-effector y-axis with position_only parameter set to true

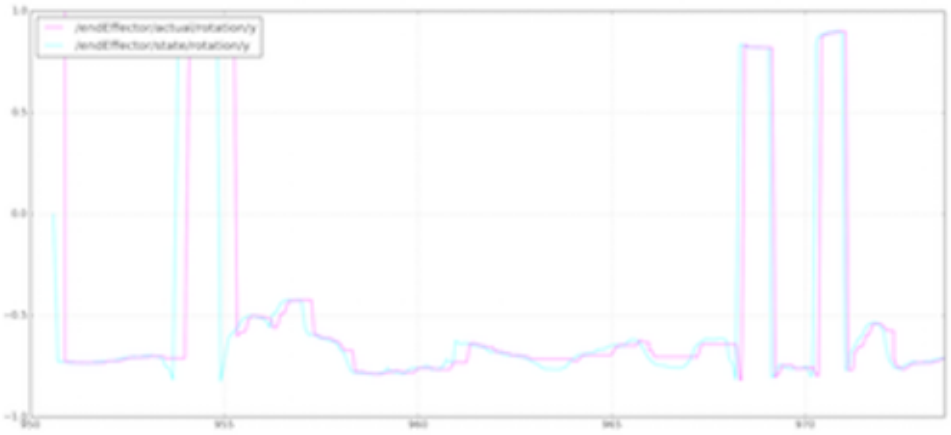


FIGURE 4.10: Rotation of end-effector y-axis with position_only parameter set to false

The results show that when the inverse kinematics solver ignores the orientation, the actual position of the end effector accurately matches the requested position (Fig. 4.7). However, as expected, it causes the orientation of the end effector to behave in an erratic manner (Fig. 4.9). This would result in the end effector being incapable of accurately picking up objects. The orientation of the end effector is seen to be more stable when using both position and orientation (Fig. 4.10). However, the inverse kinematics solver fails to find solutions as often for the given position (Fig. 4.8). As a result of these findings, it was decided to use

both the position and orientation of the end effector when calculating the inverse kinematics of the manipulator as the orientation improvements outweighed the position limitation. To implement this behaviour, the position_only_ik parameter was set to false.

4.4.4 Solve Type Parameter

Within the trac-IK plugin (track IK [2016*a*]), there are four different methods that can be used to solve the inverse kinematics. These methods are speed, distance, manipulation1 and manipulation2. The 'speed' solve type returns the first solution found in the fastest possible time. The 'distance' solve type returns the solution that minimises the change of the joint values required by the manipulator. The 'manipulation1' solve type returns a solution that avoids situations in which the arm would be folded up too close or stretched out too far from the base. 'Manipulation2' solve type is a variation of the manipulator1 solve type.

With the 'position only' parameter set to false, the different solve types were compared using the y-axis position of the end effector and a fixed timeout speed of 0.005 seconds. The required end effector position is shown as the blue line. This is compared to the actual end effector position which is shown as the red line. The x-axis represents the joint state value while the y-axis is time.

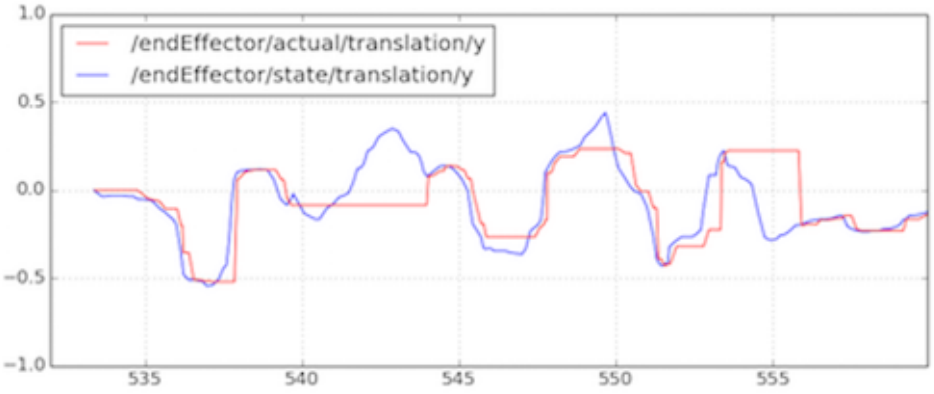


FIGURE 4.11: Solve_type: Speed

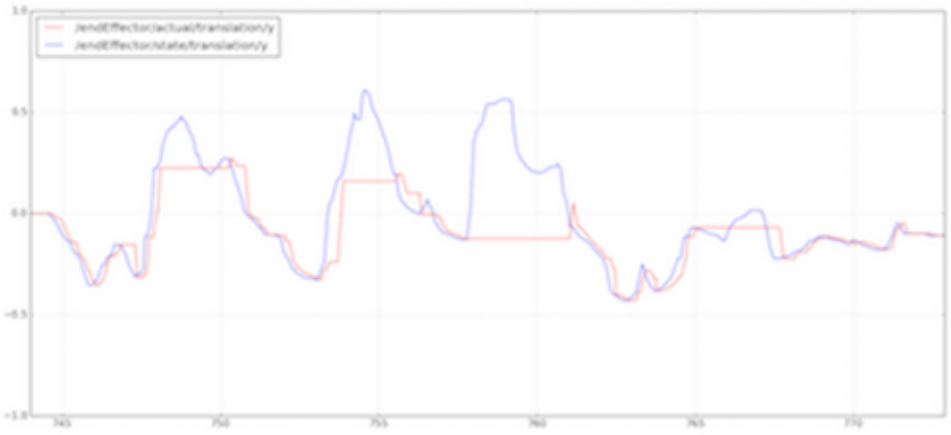


FIGURE 4.12: Solve_type: Distance

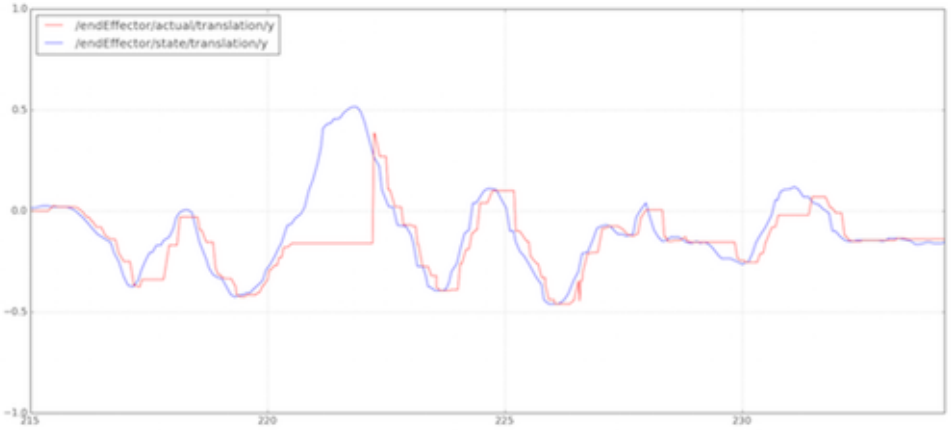


FIGURE 4.13: Solve_type: Manipulation 1

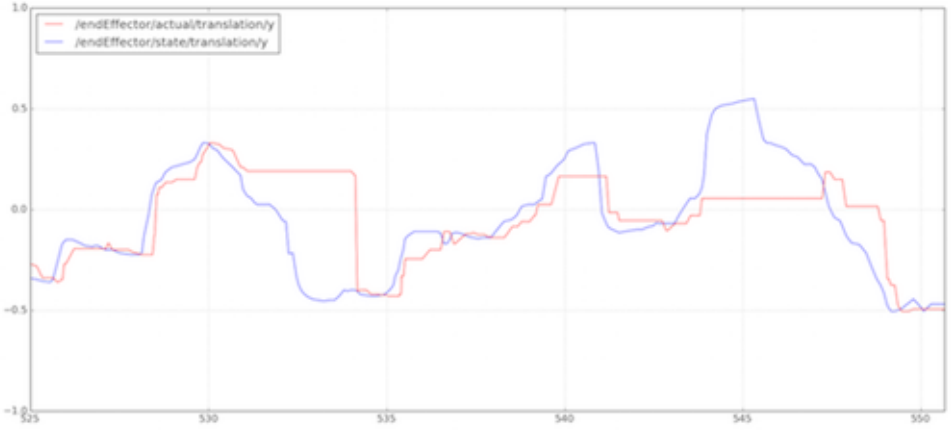


FIGURE 4.14: Solve_type: Manipulation 2

Ideally, the actual position of the end effector should be identical to the requested position. As can be seen in the graphs above, none of the solvers follow this line perfectly. However, manipulation1 solve type (Fig. 4.13) seems to behave the best when comparing requested and actual position.

4.4.5 Publishing Joint States

When requesting the inverse kinematics solution for a given position and orientation it is not always the case that a solution is found. This is due to the manipulator not having a joint state configuration that allows the end effector to reach the requested position with the given orientation. This may be due to the position being out of the manipulator workspace or having reached a singularity. In this case, no joint values are published, keeping the manipulator in the same position.

If a solution for the inverse kinematics is found, the calculated joint state values are published. These values are then used to update the manipulator joints to position the end effector. Additionally, the resulting position and the orientation of the end effector are published. The allows the system to constantly keep track of the end effector's location. This information is used when unlocking the manipulator after being paused as it allowed the operator to position their hand back into the correct position.

4.5 Stabilising Values

In the initial experiments using the simulator, both the input signals from the Leap Motion and the output signals sent to the manipulator appeared sharp and erratic as can be seen in Fig. 4.13. While an aggressive controller is not an issue in the simulator, when applied to the HDT manipulator it can cause the arm to become erratic which in turn could damage the hardware.

The controller works as an open loop system, meaning that the output does not feed back into the input to stabilise it. An open loop system was chosen as the operator tends to move their hand too quickly for closed loop system to be beneficial. It is therefore important to smooth the signals, while making sure that the operator still feels in control of the manipulator.

Damping filters were applied to both the input signals, reading the data from the Leap Motion, as well as the output signals, passing joint states to the manipulator. By filtering the input signal, small perturbations from the operator's hand were discarded by applying bound limits, and the signal noise was reduced using a mean filter. This minimised the number of unnecessary inverse kinematics calculations. The output signal was filtered to create a smoother trajectory using an α - β filter, as well removing any signal spikes using a median filter. The velocity of the manipulator was also varied to further control the trajectory of the manipulator when performing large movements.

4.5.1 Input Stabilisation

When reading the Leap Motion hand and finger signals, it is important to make sure the signals are smooth and stable. This minimises the chance of the output signal being erratic.

4.5.1.1 Bounds

When the operator holds their hand still above the Leap Motion a straight line is expected on the position graph (4.15) with the x, shown in dark blue, y, shown in red, and z, shown in light blue, position remaining fixed. As can be seen in figure 4.15, without any form of damping this is not the case. The x-axis represents the Leap Motion position value while the y-axis is time.

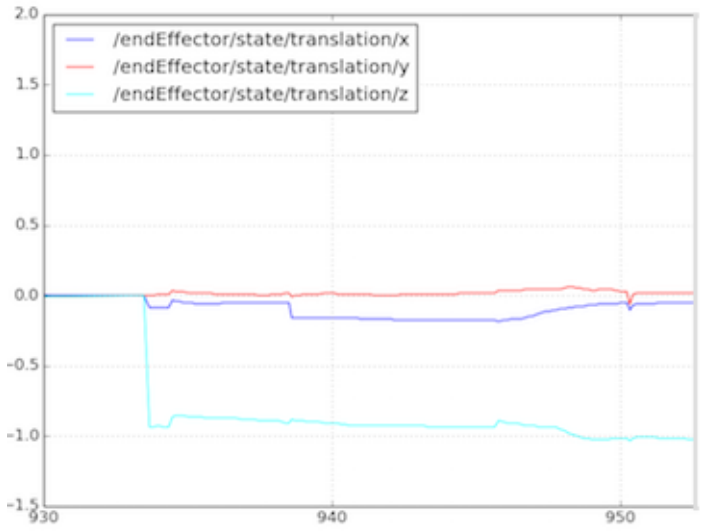


FIGURE 4.15: Leap Motion raw input

The Leap Motion is an extremely sensitive device. Natural hand tremors are detected by the Leap Motion even when the operator's hand appears stationary. Limits are set so the operator's hand appears to be at a constant position when stationary. The position is only updated when the change is more than 0.009 and the orientation is only updated when the change is more than 0.05.

4.5.1.2 Mean Filter

As well as applying bound limits on the Leap Motion input, a mean filter is also applied to the signal to further reduce noise. The mean filter is calculated by taking the mean of a sample number of values. For this controller, a small window of four is used to create smoothing while keeping the lag to a minimum.

4.5.2 Output Stabilisation

When outputting the joint values, it important that the signal is smooth, without any sudden spikes. This reduces the chance of the manipulator moving erratically and damaging itself.

4.5.2.1 Joint Limits

When the end effector moves from left to right, it eventually reaches a singularity. This means that the manipulator is unable to move the end effector further right no matter how it moves its joints. If the operator continues moving their hand to the right, the required position becomes reachable. However, to reach this position the base of the end effector has to flip past the centre point to the other side. When doing this flip, it is important that the manipulator rotation is completed in a controlled manner. It is possible to control both the position and the velocity of the joint states. The maximum velocity of the HDT manipulator is 2.6 radians/second, however to control the manipulator better the velocity was set to 0.5. When passing from left to right, some joints pass the 0 values centre line and flip from being positive to negative or vice versa. This sudden flip can be controlled by slowing down the velocity and setting the value to 0 as an intermediate point.

4.5.2.2 Alpha-Beta Filter

As explained in Jamwal [2012], the α - β filter uses estimation to apply data smoothing. The α - β filter is applied before sending the stabilised joint values to the manipulator. This filter allows the movement of the manipulator to be less erratic. The α - β filter value is calculated as follows:

Prediction equations: The predicted position and velocity of the specified joint are calculated using the previous position and velocity for a given sample time.

$$x_p(n) = x_p(n-1) + (v_p(n-1) * \delta t) \tag{4.17}$$

$$v_p(n) = v_p(n-1) \tag{4.18}$$

Error equation: The actual position and the predicted position are compared to get the error between them.

$$\epsilon = x(n) - x_p(n) \tag{4.19}$$

Smoothing equations: The newly calculated smoothed position and velocity are then calculated. The error is multiplied both to the α and β values. The α value controls the magnitude of damping with larger α values causing less damping. The β value controls surges in the velocity with low values causing less of a surge.

$$x_p(n) = x_p(n) + \alpha * \epsilon \tag{4.20}$$

$$v_p(n) = v_p(n) + (\beta * \epsilon)/\delta t \tag{4.21}$$

Update equations: The old position and velocity values are updated with the newly calculated values.

$$x_p(n-1) = x_p(n) \tag{4.22}$$

$$v_p(n-1) = v_p(n) \tag{4.23}$$

where:

$x_p(n)$: Predicted position. This value is initially set to zero.

$x_p(n-1)$: Previously predicted position.

$v_p(n)$: Predicted velocity. This value is initially set to zero.

$v_p(n-1)$: Previously predicted velocity.

$x(n)$: Current requested joint value.

ϵ: Error between predicted position and requested position.

δt: Sampling time interval. A smaller time step results in a smoother signal. This value was set as 0.5.

α: Controls the level of damping. This values was set as 0.4.

β: Controls surges in velocity. This values was set as 0.005.

4.5.2.3 Median Filter

A median filter is used to remove any spikes in the values and smooth out the trajectory of the manipulator. The median filter value is calculated by taking the median of a sample number of values. For this controller a small sample size of 5 is used to create smoothing while keeping the lag to a minimum.

4.5.3 Dynamic Reconfiguration

Most of the filters described have parameters that can be varied to provide a different output behaviour. In the case of the α-β filter the values of α, β and the sample time can be altered and in the case of both the median and mean filter the sample size can be altered. The use of a dynamic reconfigure aids in choosing the desired variable values. ROS has a package called 'dynamic_reconfigure' (ROS [2016*a*]) that allows the operator to set a number of reconfigurable variables. The dynamic reconfigure program displayed a slider for each variable. This enabled the variable values to be frequently altered while the program was running, allowing the values to be chosen once the ideal manipulator behaviour was observed.

4.6 Operator commands

4.6.1 Simulator

The simulator was used when simulating the manipulator as well as acting as an aid when controlling the real HDT manipulator. This is due to the need for feedback information being displayed on the screen. It is hoped that in the future, this information can be overlaid directly on to the real-time visual feedback from the manipulator, thereby removing the need for the simulator in this case.

4.6.2 Keyboard

The keyboard is used for both starting and pausing the controller. Initially, it had been hoped that the operator could use the detection of their left hand to control the pause mechanism of the system. Unfortunately, the current version of the Leap Motion software does not deal well with occlusion. Occlusion occurs when the operator's hands are overlapped, making it difficult for the Leap Motion to differentiate between the two hands. As a result, the left hand was not recognised consistently enough for this to be a plausible method. Instead, a keyboard press (ROS [2016*b*]) was used.

4.6.3 Starting the Program

To give the operator time to position their hand above the Leap Motion, the system starts only once the 's' key is pressed. Once pressed the operator has full control of the manipulator.

4.6.4 Pausing and Resuming the System

It was found in the literature to be important to have a mechanism that allows the operator to pause the system. Without this, the operator has no way of resting their hand and arm. This would result in the operator becoming tired after controlling the system for long periods of time.

4.6.4.1 Pause

To pause the system, the operator presses the 'space' key. This causes the manipulator to remain fixed at its current position. This allows the operator to remove their hand and step away from the computer. A message is displayed on the RViz simulator informing the operator that the system is now paused.

4.6.4.2 Lock

The system is resumed by pressing the 'space' key. Once the system is resumed it is important for the operator to have their hand in the correct position in relative space. This prevents the manipulator from being damage due to it jumping rapidly to another location. Additionally, without this, the operator's hand would no longer be synchronised with the manipulator's end effector position. To help the operator position their hand correctly, the system remains locked until the operator is within a range of $\pm$ 0.07 meters from the end-effector's last valid position. A smaller range would make it hard for the operator to position their hand correctly, while a larger range would increase the chances of the manipulator jumping. A coloured sphere appears within the RViz simulator, giving the operator a visual guide as to where to place their hand. A message is displayed to the operator within the RViz simulation informing them that the program is locked and that they have to place their hand over the sphere to unlock. Once the operator's hand is within the sphere the system is unlocked, allowing the operator to continue teleoperating the system.

Chapter 5

Findings

5.1 Introduction

Timed trials and a questionnaire were used to test the stated hypothesis. This hypothesis aimed to prove that a gesture control input would provide a more intuitive method of control compared to the HOCU controller.

Each operator had to try and complete a set task, which was repeated three times, in the shortest possible time. After completing the three tries, the operator then had to complete a questionnaire . All operators completed this process on both the Leap Motion controller and the HOCU. The task and the questionnaire were designed to explore how the Leap Motion controller compared to the HOCU in terms of intuitiveness, manoeuvrability, ease of use and comfort. Once all the data was recorded, the averages were taken for all timed trials on each device and the scores for the questions were averaged for each device.

A copy of the questionnaire can be found in Appendix A, the individual operator questionnaire responses can be found in Appendix B, the operation instructions given to each operator can be found in Appendix C, while the individual trials times can be found in Appendix D.

5.2 Task

The task set for each trial was to hit a buoy to the left of the manipulator, wait for 10 seconds while remaining in contact with the buoy and then proceed to hit the other buoy to the right of the manipulator. The elapsed time to hit the right buoy and the elapsed time to complete the full task were measured. The cut-off time for completing the whole task was set to 5 minutes 30 seconds.This time was chosen as a result of a number of tests. Any trial where the operator did not complete the task within the cut off time was viewed as a failed attempt.

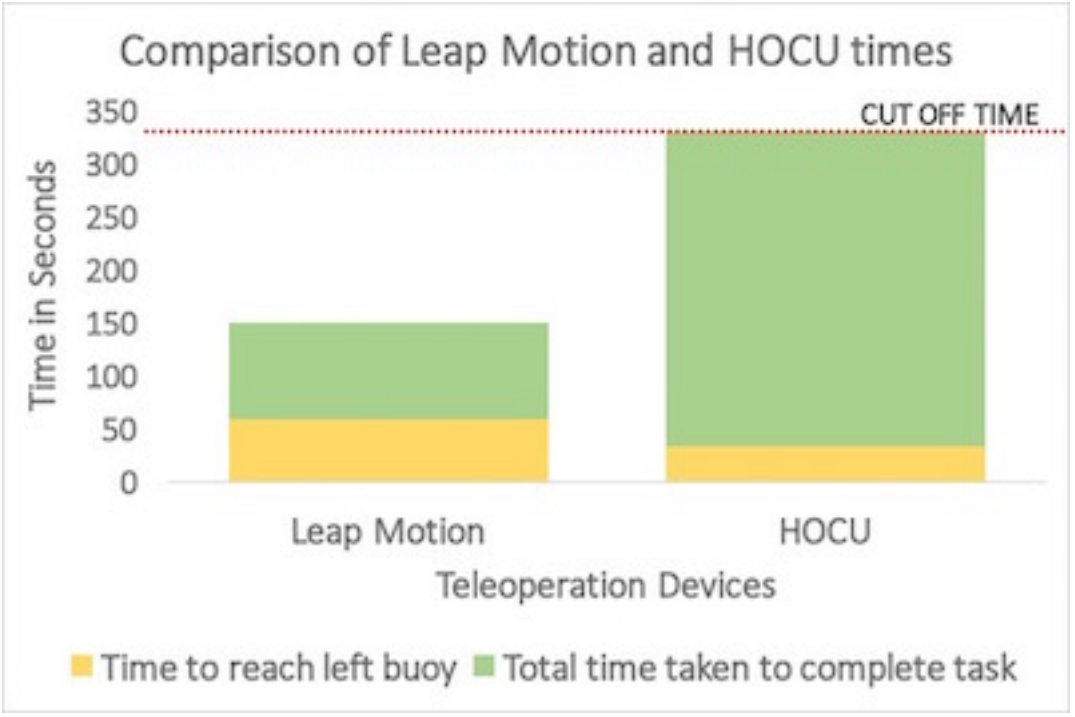


FIGURE 5.1: Results of the timed trials

As can be seen from the graph, when using the Leap Motion system, the operators managed to complete the task nearly every time. While using the HOCU, none of the operators managed to complete the task. When trying to reach the right buoy the manipulator consistently hit joint limits when using the HOCU, making the manipulator hard to control. This caused the manipulator to become

erratic. This occasionally resulted in the manipulator having to be restarted as there was a risk of the manipulator damaging itself.

The manipulator drivers would often disconnect themselves, only reconnecting once the manipulator had been restarted. The cause of this issue was traced back to a loose connection that was disconnected when the manipulator moved erratically. The above issues limited the ability to collect reliable experimental data for the HOCU. The Leap motion got round the issue of joint limits by ignoring them and remained in a fixed position until the operator specified a position that the manipulator could reach.

While the HOCU failed to complete the task, it may be the case that this was due to software faults rather than hardware. Had the software been able to deal with joint limits better it may have proved to be as successful or more so than the Leap Motion when completing the timed trials. However, the HOCU hardware was still the cause of a number of issues addressed in the questionnaire.

5.3 Questionnaire

As well as the time taken to complete the task, it was important to gauge the operators feeling of intuitiveness, ease of use and comfort. This information was collected using a questionnaire. The questionnaire asked five questions which checked how intuitive the device was, how easy it was to take a break, how painful it was to use it, the device precision and control. Operators were asked to give each question a rating between 1 and 5, where 1 indicated a low response to the question.

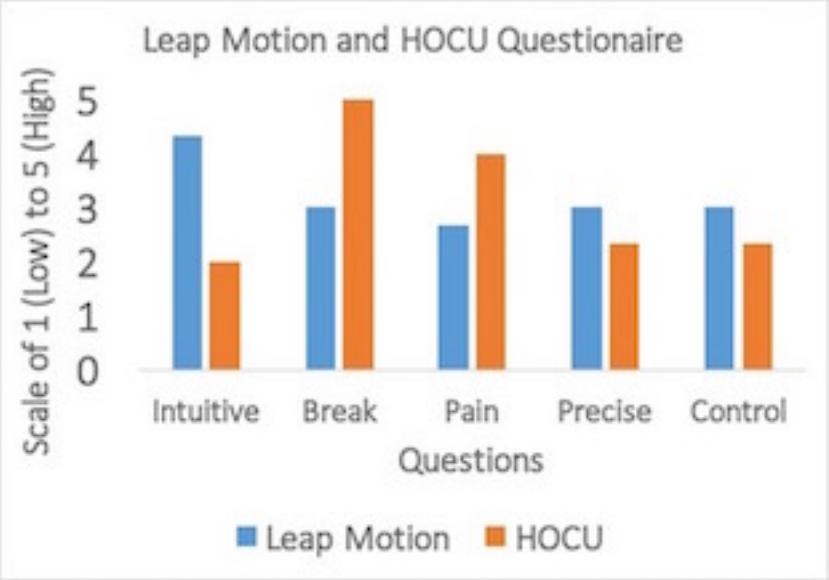


FIGURE 5.2: Results from the questionnaire

Intuitiveness

All three participants found the Leap Motion to be more intuitive than the HOCU controller. The Leap Motion and manipulator positions were synchronised, allowing the user to move their hand left and right and the manipulator to follow. As the operator's hand is fixed to the joystick, it was hard to move in three dimensions intuitively.

Pausing

The only point when the Leap Motion lagged compared to the HOCU was when pausing. To pause the HOCU, the operator can just let go of the joystick. However, as expected, this would not work using the Leap Motion as the Leap Motion is controlled using relative hand position. Currently, the pause operation is controlled by the space key on the keyboard. However, it is hoped that with the Orion improvements developed for the Leap Motion in future works the detection of the operator's left hand can be used to pause the system. This would allow the operator to remain focused on keeping their hand steady while pausing. This

removes the need for the operator to shift their concentration onto the keyboard to find the space key.

Discomfort

In general, users found that using the HOCU caused them more pain than when using the Leap Motion. To move the manipulator using the HOCU, the operator must push and pull the joystick with reasonable force. This causes the operator's hand to become strained after using the HOCU for a while. Although the results indicate that the Leap Motion did not cause as much pain as HOCU, some pain was felt due to holding their hand above the Leap Motion for an extended period.

Precision

On average, the Leap Motion was found to be more precise. The Leap Motion gives sensitive readings and this allows the operator to make small, precise adjustments to the manipulator position.

Control

The findings indicate that relative to HOCU, the Leap Motion allows the operators to feel more in control of the manipulator. During the trials, when the HOCU hit joint limits, it would often act in an unpredictable and erratic manner. This made it hard for the operators to feel in control of the manipulator as the manipulator often behaved differently to what the operators expected.

Chapter 6

Conclusion

6.1 Introduction

This research aimed to find a stable and intuitive way of interacting and controlling an underwater manipulator during teleoperation. The research findings suggest that using a gesture control input device, such as the Leap Motion instead of the HOCU, is a viable option. This chapter summarises the findings and discusses possibilities for future work.

6.2 Summary of Findings and Conclusion

A literature review of this subject area revealed that there are many different input control mechanisms. Examples of these include data gloves, stereo cameras and depth aware cameras. Following an evaluation of the different controllers, gesture control was found to be the most suitable form of input for teleoperations. Amongst the different available gesture control devices, the most popular technologies appear to be the Leap Motion and the Kinect. For this research, the Leap Motion was seen as the better option. This was due to the Kinect's lack of finger tracking which meant that an additional device would have to be used to control the end effector's fingers and thumb.

There a number of challenges that need to be considered when designing a teleoperation controller using the Leap Motion. For example, the initial calibration of the manipulator has to be set in order for the operator and the manipulator to have similar starting positions. This allows for a more natural controller. Additionally, to get the required joint state values, inverse kinematics has to be applied on the requested end effector position. Before these joint state values can be sent to the HDT manipulator they have to be stabilised. This is important as the manipulator trajectory has to be smooth to minimise the risk of damaging the hardware. Lastly, it is important that when unlocking the system the operator position their hand such that it is in the same relative 3 dimensional space as the manipulator.

It was found that relative to the HOCU controller, the Leap Motion device provides a more intuitive method of controlling the HDT Adroit Underwater Manipulator. Using a gesture controlled device creates a more natural and intuitive teleoperation system, which improves the operator's control of the manipulator. This finding is based on a number of experiments using simulation and a manipulator with no attached end-effector.

6.3 Recommendations

While the research findings proved to be a success, further improvements to allow for a more natural method of operation known as telepresence can be pursued. Telepresence relies on the operator receiving multiple inputs, such as visual and tactile feedback. This enables the operator to feel as though they are at the operation site, resulting in a more intuitive controller.

6.3.1 Software Development

While gesture control has come a long way in the past few years, there is still extensive research being conducted to improve the controller's limitations. One

of the main limitations with the Leap Motion is its ability to deal with hand occlusion. A new version of the Leap Motion software, which they have named Orion, has recently been developed. Within Orion, it is stated that the occlusion issues are fixed, however currently this software is only available on the Windows operating system.

6.3.2 Visual Feedback

6.3.2.1 Virtual Reality Headset

Recently there has been a lot of focus on virtual reality (VR) headsets such as the Oculus Rift. Both the Kinect and Leap Motion software are now compatible with such headsets. These head-mounted displays (HMD) allow the operator to view the environment from the perspective of an external camera, mounted on the hardware. This camera can then be controlled using the movement of the operator's head resulting in an more natural control system (Almeida et al. [2014]). In addition, it allows the operator to feel as if they are physically located at the site, a feeling known as *embodiment*.

6.3.2.2 Embodiment

Physical embodiment in teleoperation helps improve the performance of the operator's task (Almeida et al. [2014]) as it improves the ease of use. Physical embodiment relies on the operator perceiving synchronous actions and sensations.

In order to make sure the system is synchronous, special focus is needed on the lag of the system. If there is too much lag the movements stop feeling natural and the sense of ownership within embodiment is effected (Aymerich-Franch et al. [2015]). The lag time that the hardware can have, while still feeling natural, needs to be investigated.

Additionally, when looking at a free hand in space it is hard for the operator to associate with the free hand as being part of their body. However, it may be

the case that by overlaying a body around the manipulator the operator will feel more connected to it.

6.3.2.3 Camera

When using the movement of the operator's head to control a camera, three main topics need to be considered:

- **Camera position:** For the operator to experience the feeling of embodiment the camera needs to be in the same position as the operator's head would be relative to the manipulator. As a result, the operator would feel as though they were looking at their arm and hand. If the camera is in the wrong position, the feeling of embodiment would be lost as what the operator saw would not line up with their normal view of their body.
- **Camera control:** As well as the position of the camera, the movement of the camera also needs to be considered. The human head is capable of a wide range of movements. Ideally, a camera is required that is capable of performing the same movements as the operator's head.
- **Balance:** Another thing to consider is balance. Once a camera is attached, the centre of balance of the manipulator is likely to change. Depending on the size of the camera relative to the manipulator there is a chance that this will affect the control of the manipulator.

6.3.3 Tactile Feedback

6.3.3.1 Haptic Devices

One of the leading fields of research which would allow for tactile feedback is the development of haptic devices. Tactile feedback would allow the operator to feel as if they were directly touching the object that the end effector was grasping. This would allow the operator to adjust the strength of their grip appropriately.

There are many different variations of haptic feedback devices that are currently being developed that can compliment gesture control devices. These include devices that can be clipped on the operator's fingers, which use pistons or expandable air pockets to apply force on to the operator's fingers. Another new development is the use of ultrasound waves to provide mid-air haptic. While these devices are still being developed it is hoped that they would be a viable option in the near future.

6.3.3.2 Touch Sensors

As well as providing feedback to the operator, the end effector needs to receive feedback from the object it is grasping. There are many sensing devices available such as capacitive sensors, strain gauge sensors and piezoelectric sensors. Providing a sense of touch would significantly improve the grasping capabilities of the end-effector.

6.4 Contribution to Knowledge

Currently available teleoperation controllers are often unnatural and unintuitive to use and require extensive training. This research has shown that a gesture control based input is a natural and intuitive means of teleoperation. Controllers that allow operators to use a more natural interaction to control underwater manipulators will save time and money by eliminating the required operator training and will enable a larger number of operators to control underwater manipulators. With advances in research and development of tracking software, virtual reality and haptics, we are now on the verge of creating the feeling of embodiment within teleoperation. This should deliver a more intuitive means of telepresence for underwater manipulators.

Appendices

A Questionnaire

On a scale of 1 (Low) to 5 (High) :

- Question 1: How intuitive did you find the device
- Question 2: How easy did you find taking a break from using the device
- Question 3: How tired or sore did you feel after using the device
- Question 4: How precise did you feel the device was when controlling the manipulator
- Question 5: How in control of the manipulator did you feel when using the device

B Questionnaire Results

On a scale of 1 (Low) to 5 (High) :

User 1	Q1. Intuitive	Q2. Break	Q3. Pain	Q4. Precise	Q5. Control
Leap Motion	4	4	3	3	2
HOCU	2	5	3	4	4

User 2	Q1. Intuitive	Q2. Break	Q3. Pain	Q4. Precise	Q5. Control
Leap Motion	4	2	3	3	4
HOCU	2	5	5	1	1

User 3	Q1. Intuitive	Q2. Break	Q3. Pain	Q4. Precise	Q5. Control
Leap Motion	5	3	2	3	3
HOCU	2	5	4	2	2

C Operation Instructions

Leap Motion

Start

- Hold your right hand directly above the Leap Motion device with your palm facing downwards. Your hand should be around 20 cm above the sensor.
- When you are ready press the 's' key on the keyboard keeping your hand still
- Once the leap motion axis appears on the RViz simulation you can then move your hand to control the manipulator

Pause

- Pause and resume are controlled by toggling the space key
- Instructions on how to gain control of the manipulator after pausing it are shown on the RViz simulation screen

HOCU

Start

- The start button is pressed (See HOCU screen for button names)

Home

- The home button resets the manipulator back to its original position

Once the arm is set to home, the joystick can be held. In order to move the arm the button on the joystick must be held in .

D Trial Times

User 1				
		Trial 1	Trial 2	Trial 3
	Right Bouy	D.N.T	D.N.T	D.N.T
Leap Motion	Total time	P.C.T	2 minutes 4 seconds	48 seconds
	Right Bouy	D.N.T	D.N.T	D.N.T
HOCU	Total time	P.C.T	P.C.T	P.C.T

User 2				
		Trial 1	Trial 2	Trial 3
	Right Bouy	1 minutes 29 seconds	21 seconds / 20 seconds	4 seconds
HOCU	Total time	P.C.T	D.N.C	D.N.C
	Right Bouy	1 minutes 25 seconds	2 minute 17 seconds	1 minute 16 seconds
Leap Motion	Total time	3 minutes 40 seconds	3 minutes 25 seconds	3 minute 58 seconds

User 3				
		Trial 1	Trial 2	Trial 3
	Right Bouy	24 seconds	6 seconds	22 secodns
Leap Motion	Total time	1 minutes 4 seconds	36 seconds	48 seconds
	Right Bouy	D.N.C	D.N.C	D.N.C
HOCU	Total time	D.N.C	D.N.C	D.N.C