

NOVEL TELE-OPERATION OF MOBILE-MANIPULATOR SYSTEMS

by

Michael C. Frejek

BEng (Hons), University of Ontario Institute of Technology, 2007

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

The Faculty of Engineering and Applied Science

Mechanical Engineering

Supervisor: Scott Nokleby, Faculty of Engineering and Applied Science
Examining Board: Remon Pop-Iliev, Faculty of Engineering and Applied Science
External Examiner: Ed Waller, Faculty of Energy Systems and Nuclear Science

THE UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

August 2009

©Michael C. Frejek 2009

Abstract

A novel algorithm for the simplified tele-operation of mobile-manipulator systems is presented. The algorithm allows for unified, intuitive, and coordinated control of mobile manipulators, systems comprised of a robotic arm mounted on a mobile base. Unlike other approaches, the mobile-manipulator system is modeled and controlled as two separate entities rather than as a whole. The algorithm consists of three states. In the first state a 6-DOF (degree-of-freedom) joystick is used to freely control the manipulator's position and orientation. The second state occurs when the manipulator approaches a singular configuration, a configuration where the arm instantaneously loses a DOF of motion capability. This state causes the mobile base to proceed in such a way as to keep the end-effector moving in its last direction of motion. This is done through the use of a constrained optimization routine. The third state is triggered by the user: once the end-effector is in the desired position, the mobile base and manipulator both move with respect to one another keeping the end-effector stationary and placing the manipulator into an ideal configuration. The proposed algorithm avoids the problems of algorithmic singularities and simplifies the control approach. The algorithm has been implemented on the Jasper Mobile-Manipulator System. Test results show that the developed algorithm is effective at moving the system in an intuitive manner.

Dedication

To my family and friends.

Acknowledgments

First and foremost, I would like to thank Dr. Scott Nokleby for all his support. He has been there every step of the way to guide me through my research, to answer all my questions, and to provide encouragement when things seemed grim. I could have not asked for a better supervisor.

I would like to thank my family. If it were not for my father and mother instilling in me the value of education and hard work, I would not be writing this thesis.

I cannot forget to mention my fellow colleagues and friends. Their fellowship was invaluable.

Lastly, I must also thank Cameco Corporation (through the Cameco Research Chair in Nuclear Fuel at UOIT), the Natural Sciences and Engineering Research Council (NSERC) of Canada, and the Canada Foundation for Innovation (CFI) for providing funding to support this research. Without their help, this would not be possible.

Table of Contents

Dedication	i
Acknowledgments	ii
Table of Contents	iii
List of Tables	vi
List of Figures	vii
1 Introduction	1
1.1 Background	2
1.1.0.1 Advantages	4
1.1.0.2 Disadvantages	7
1.1.1 Current State of the Art	8
1.1.2 Literature Review	15
1.1.3 Problem Statement and Goal	22
1.2 Scope	24
1.3 Thesis Outline	25
2 Hardware	26
2.1 Manipulator	27
2.2 Mobile Base	28

2.3	Joystick	30
2.4	Operations Computer	34
2.5	Overall System	35
3	Mobile-Manipulator Kinematics	37
3.1	Manipulator	37
3.1.1	Zero Displacement Diagram	38
3.1.2	Denavit and Hartenberg Parameters	39
3.1.3	Forward Displacement Solution of the Wrist Centre	40
3.1.4	Inverse Displacement Solution of the Wrist Centre	41
3.1.5	Jacobian of the Wrist Centre	43
3.2	Singularities of the Manipulator	45
3.3	Mobile Base	46
4	Control Algorithm	49
4.1	Design Process	49
4.1.1	Concept Generation	53
4.1.1.1	Input Device	53
4.1.1.2	Manipulator Control	54
4.1.1.3	Mobile Base Control/Unification	61
4.2	Overview	63
4.3	State 1: Manipulation	65
4.4	State 2: Approaching a Singularity	72
4.4.1	Pre-Optimization	73
4.4.2	Optimization	75
4.4.2.1	Objective Function	76
4.4.2.2	Constraints	76
4.4.3	Post-Optimization	83

4.5	State 3: Ideal Configuration	85
4.5.1	Pre-Optimization	87
4.5.2	Optimization	88
4.5.2.1	Objective Function	88
4.5.2.2	Constraints	89
4.5.3	Post-Optimization	91
4.6	Summary	92
5	Implementation	94
5.1	System Architecture	94
5.2	Initialization	96
5.3	Joystick Program	98
5.4	Controller Program	102
5.5	State 2/State 3 Optimization Programs	108
5.6	Distributor Program	110
5.7	Manipulator Program	112
5.8	Mobile Base Program	122
5.9	Summary	125
6	Results and Discussion	127
6.1	Motion Tests	127
6.1.1	State 1	128
6.1.2	State 2	128
6.1.3	State 3	132
6.2	Application Test	133
6.3	Sources of Error	135
7	Conclusions and Recommendations for Future Work	137
7.1	Conclusions	137

7.2	Recommendations for Future Work	138
-----	---	-----

List of Tables

2.1	Thermo Fisher Scientific CRS F3 Arm Specifications [41]	28
2.2	Thermo Fisher Scientific CRS C500C Controller Specifications [41] . .	29
2.3	PowerBot TM AGV's Specifications [44]	31
2.4	Operations Computer Specifications [41]	35
3.1	Denavit and Hartenberg Parameters for the CRS F3 Manipulator . .	40
4.1	CRS F3 Joint Constraints [41]	81

List of Figures

1.1	Manipulator	2
1.2	Mobile Robot	2
1.3	Mobile Manipulator	3
1.4	Six Degrees-of-Freedom	4
1.5	Redundancy	5
1.6	Redundancy Exploitation: Force	6
1.7	Redundancy Exploitation: Manipulability	6
1.8	Mobile Manipulator used for Forging [5]	9
1.9	Mobile Manipulator used for Lifting and Maneuvering Loads [6]	10
1.10	Mobile Manipulator for Removing Coating from Aircraft [7]	11
1.11	Mobile Manipulator used for Construction Work [8]	11
1.12	Small-Scale Industrial Mobile Manipulator [9]	12
1.13	tEODor Mobile-Manipulator System [10]	12
1.14	Packbots [11]	13
1.15	Warrior Mobile Manipulator [11]	13
1.16	Shark Marine Sea-Dragon ROV [12]	14
1.17	Saab Seaeye Panther ROV [13]	14
1.18	Mobile Servicing System [14]	15
1.19	Traditional Control of Mobile Manipulators	23
2.1	Jasper Mobile-Manipulator System	27

2.2	Joints of the CRS F3 Articulated Arm	28
2.3	PowerBot™ AGV [43]	29
2.4	Six Degree-of-Freedom Spatial Joystick	32
2.5	Six Degree-of-Freedom Spatial Joystick	33
2.6	Conversion Box	34
2.7	Overall System Architecture	36
3.1	Zero Displacement Diagram of the F3 Manipulator	38
3.2	Singular Configuration: $s_5 = 0$	46
3.3	Singular Configuration: $c_3 = 0$	46
3.4	Mobile Base	47
4.1	Control using World Coordinates	56
4.2	Control using Tool Coordinates	57
4.3	Control States	64
4.4	State 1	65
4.5	Control Example for State 1	66
4.6	Joystick Frames	67
4.7	Manipulator Frames	68
4.8	Mapping of Joystick Frame to Tool Frame	68
4.9	Position Deadband	71
4.10	Orientation Deadband	71
4.11	State 2	72
4.12	Wrist Centre Position Determination	74
4.13	Determination of New Wrist Centre Position	80
4.14	Maximum Possible Joint Movement	82
4.15	Precedence of Maximum Joint Range	82
4.16	Error in End-Effector Orientation	83

4.17	Frames of Mobile Manipulator	86
4.18	State 3	86
4.19	Ideal Pose	88
4.20	State 3 without Adjusting $\theta_4, \theta_5, \theta_6$	91
5.1	System Architecture	95
5.2	Initialization Sequence	97
5.3	Joystick Program Overview	101
5.4	Controller Program Overview	104
5.5	Controller Program Overview Cont'd.	105
5.6	Position Deadband Filter Overview	106
5.7	State 2 and State 3 Optimization Program Overview	109
5.8	Distributor Program Overview	111
5.9	Manipulator Program Overview	114
5.10	Manipulator Program Overview Cont'd.	115
5.11	Manipulator Program Overview Cont'd.	116
5.12	Mobile Base Program Overview	124
6.1	State 2: Last End-Effector Direction 0° (Side View)	129
6.2	State 2: Last End-Effector Direction 0° (Front View)	130
6.3	State 2: Last End-Effector Direction 45° (Side View)	130
6.4	State 2: Last End-Effector Direction 45° (Front View)	130
6.5	State 2: Last End-Effector Direction 90° (Side View)	131
6.6	State 2: Last End-Effector Direction 90° (Front View)	131
6.7	State 2: Last End-Effector Direction 180°	132
6.8	State 3: Results	133
6.9	Application Test	133
6.10	Application Test Results: State 1	134

6.11 Application Test Results: State 2	135
6.12 Application Test Results: State 3	135

Chapter 1

Introduction

For the past few decades researchers have been trying to find the best way to control mobile-manipulator systems. These systems allow for robot manipulators to be used anywhere and take the benefits of manipulators into the field. Much effort has been focused on the movement of the overall mobile-manipulator systems, with very little attention having been given to the operator. Moreover, current control strategies rely on two input devices, making it fairly difficult for an operator to control. Formulation of these control algorithms also usually leads to restrictions, both physical and algorithmic.

Hence, a new control algorithm has been proposed for the control of mobile manipulators. It has been designed and developed to overcome the current problems by placing focus on the operator. It uses one input device to control the mobile manipulator, making it easy for the operator to use. It is intuitive in nature with the manipulator and mobile base moving when necessary.

Proper design methodology has been used to develop a system that achieves the desired goals of the project.

1.1 Background

The industrial robot, also known as a manipulator or robotic arm, dates back to the mid-1950s. Figure 1.1 shows an example. It is a device comprised of links connected in a serial fashion by motor-driven joints. The joints either translate or rotate the links allowing the tool, called the end-effector, to perform precise movements [1]. From their inception, manipulators have revolutionized the industrial world, being used to complete tasks that are repetitive, need high accuracy, require speed, and/or are too dangerous for humans to perform.

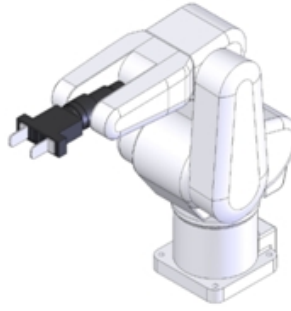


Figure 1.1: Manipulator

A mobile robot is a vehicle containing a computer and sensors [2]. Its mobility can be provided by wheels, tracks, or other forms of propulsion. Mobile robots can be used for many different tasks such as transporting objects, mapping a room, and collecting data. An example of a mobile robot is shown in Figure 1.2.

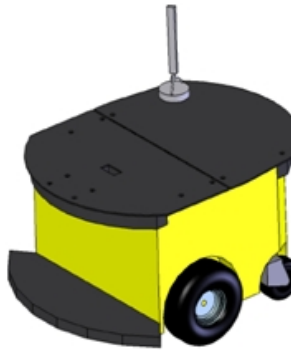


Figure 1.2: Mobile Robot

A mobile-manipulator system is a combination of the above two systems, a manipulator mounted on a mobile-robotic base, as shown in Figure 1.3. This combination has a synergistic effect. Both systems are now able to do more together than they could alone.

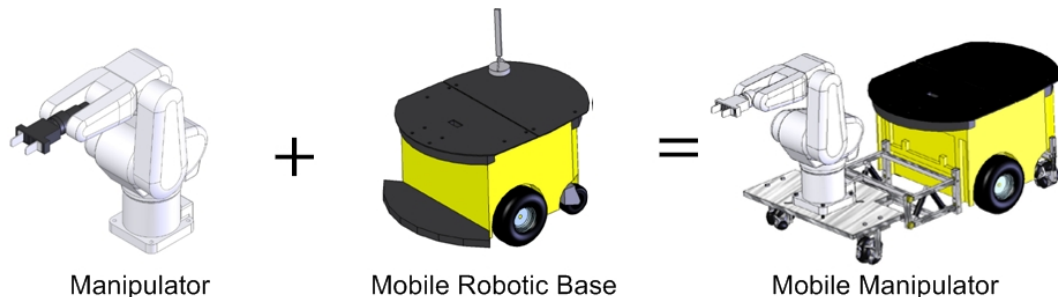


Figure 1.3: Mobile Manipulator

Tele-operation is the process of operating a device at a distance [2]. The method of input/control is usually a joystick, but can also be a computer. Data from the input device is transmitted either through wires or wirelessly depending on the environment/situation to the device. This data is then used to control the device, which can be a manipulator, a mobile robotic base, or in this case a mobile-manipulator system. A manipulator and mobile base both have a set amount of degrees-of-freedom (DOF). The number of DOF is the “the number of independent position variables that would have to be specified in order to locate all parts of the mechanism” [1]. A manipulator with 6-DOF is able to position and orient an object anywhere in space, within its workspace. It requires 3-DOF for translation and 3-DOF for orientation, as illustrated in Figure 1.4. A mobile base with 2-DOF is able to traverse an entire 2-D plane. The number of DOF of each system is specific and sufficient for each of their associated tasks. A mobile-manipulator system has extra DOF, not necessarily equal to the sum of the DOF of the manipulator and the mobile base. If one denotes that m coordinates are necessary to define the position and orientation of the end-effector and n coordinates are necessary to define the configuration of the mobile manipulator, then the mobile manipulator has $n - m$ extra DOF. Extra DOF make a mobile manipulator

redundant, meaning that it is now able to perform the same task in many different ways.

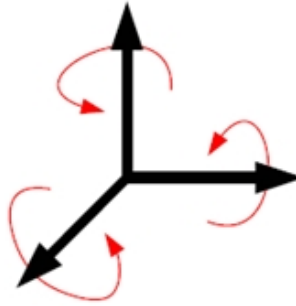


Figure 1.4: Six Degrees-of-Freedom

The added mobility makes a mobile manipulator advantageous over its stationary counterpart for several reasons which are highlighted below.

1.1.0.1 Advantages

- **Increased Workspace:** Unlike a standard manipulator, a mobile manipulator is not fixed to one location. The manipulator is able to basically go anywhere its mobile base can take it. This extends the manipulator's work envelope to the entire travel region of the mobile base.
- **Cost Reduction:** With an increased work envelope, it is possible to have the mobile manipulator perform tasks in more than one region, hence reducing the need for two or more (fixed) manipulators. With manipulators costing in the tens of thousands of dollars, this is a real cost savings.
- **Redundancy Exploitation:** For many applications, the total DOF of the mobile-manipulator system is greater than the total DOF of the task. "This means that, in principle, no manipulator is inherently redundant; rather there are certain tasks with respect to which it may become redundant" [3]. For example, consider a 2-DOF manipulator. One could place the end-effector at any position in the 2D plane (within the manipulator's workspace) with up to

two configurations of the manipulator. However, if the 2-DOF manipulator is placed on a wheeled base, the system can place the end-effector in the same position with an infinite number of configurations. This is shown in Figure 1.5. Configuration A shows a fixed manipulator with its end-effector in a set location, P. There are only two possible configurations to achieve the desired position, P. If the manipulator is placed on a wheeled base, like in Configuration B, then this same end-effector position can be achieved using an infinity of configurations. This is referred to as redundancy.

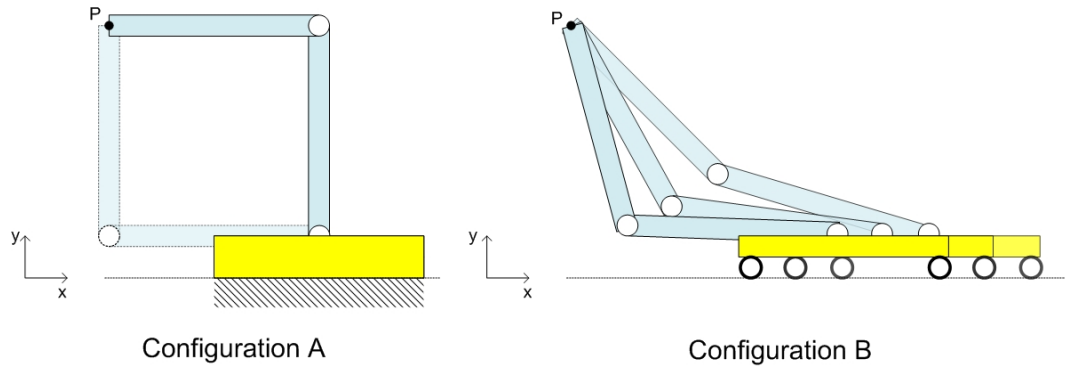


Figure 1.5: Redundancy

If one can achieve a task in an infinite number of ways, a certain way might be more beneficial. For example, a mobile manipulator's redundancy can be exploited to increase its force capabilities. In Figure 1.6, a horizontal force, $\mathbf{F}$, is applied to the end-effector. Configuration B is better at resisting the force as opposed to Configuration A. This is because the sustainable force of Configuration A is dependent on the amount of torque that joint 2 can apply, which is limited. Configuration B is dependent only on the structural strength of the outstretched manipulator arm, which is much higher. This is due to the fact that the force intersects all of the joint axes.

The redundancy can also be exploited to increase the manipulator's manipulability. Figure 1.7 shows two configurations of the mobile manipulator. The

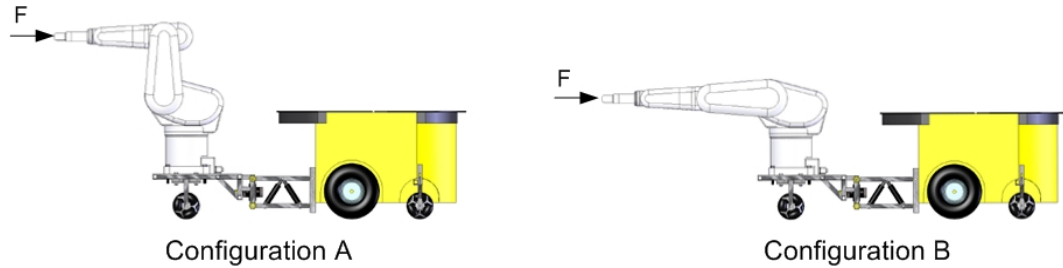


Figure 1.6: Redundancy Exploitation: Force

arrows at the end-effector symbolize the manipulator's possible movements. In Configuration A, the manipulator is outstretched and cannot go in the forward direction. In Configuration B, the manipulator is in a configuration in which it can move in all directions. Therefore, it has higher manipulability as it can operate on an object in all directions.

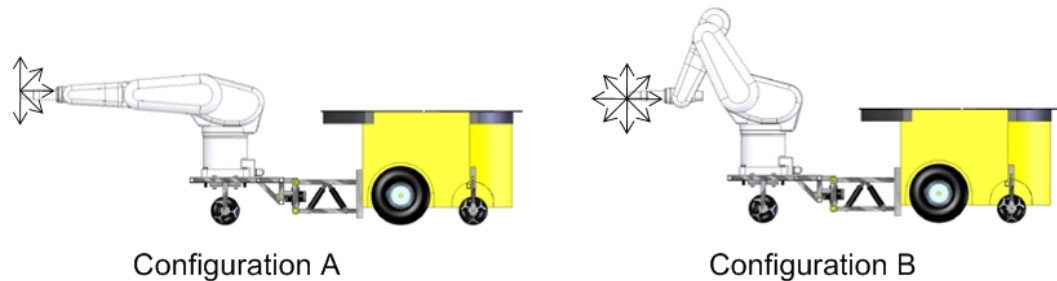


Figure 1.7: Redundancy Exploitation: Manipulability

This redundancy can also be exploited to do such things as avoid obstacles, lower the joint velocities and torques, and avoid singularities. The added flexibility that the redundancy adds makes them attractive for complex tasks.

- **Expanded Usage:** Most manipulators have been used either in an industrial or laboratory setting, due to the fact that they need to be fixed to a solid base. However, a manipulator on a mobile base allows for a manipulator to be used virtually anywhere, especially outside. This changes a manipulator's role from a manufacturing assistant to a multi-purpose assistant and allows for mobile manipulators to do such tasks as mining, logging, toxic waste disposal, and

even bomb defusing.

These added benefits do not come without any disadvantages. These are outlined below.

1.1.0.2 Disadvantages

- **Redundancy Resolution:** When it comes to control, it is much easier to control a non-redundant system as opposed to a redundant one. In controlling physical objects, the DOF of the controlling device are usually equivalent to the DOF of the actuator (i.e., an automobile has 2-DOF, one controlled by the gas pedal and the other controlled by the steering wheel). At most, input devices have 6-DOF. In a redundant system (one with more than 6-DOF), more than one input device is therefore required. This can get very complicated, especially if a human operator is controlling the device. Many real-world mobile manipulators are controlled by an operator with two inputs, one input controlling the manipulator and the other controlling the mobile base. This requires great coordination by the operator. For example, in order to move the end-effector forward, the manipulator can be instructed to move forward, the mobile base can be instructed to move forward, or a combination of the two. Herein, lies the motivation of this thesis: to develop an intuitive controller for mobile-manipulator systems to simplify the operator's job and thereby increase his/her performance.

Redundancy in the system creates an infinity of possible solutions, in this case configurations of the mobile manipulator. This has many benefits. However, this now creates the problem of choosing which solution is the best solution. Some problems may involve the need for a solution of high dexterity or a solution that provides the manipulator with the most resistive force or even a solution that requires both. Determining such solutions is difficult.

- **Vibrations:** A traveling mobile base is likely to encounter holes, protrusions, uneven terrain, as well as abrupt changes in speed and acceleration when it starts moving, turns, or stops along its path. All of these are likely to cause the mobile base to vibrate. These sudden movements of the mobile base will undoubtedly transfer to an attached manipulator causing it to lose accuracy.

These vibrations could also be caused by the manipulator as well. If big enough, a manipulator's quick movement could affect the position/motion of its mobile base.

- **Loss of Stability:** In order to be useful, the manipulator may need to be fairly long. This in turn means that the manipulator may weigh a large amount. Therefore, the possibility exists that if the arm is outstretched or if it is carrying a heavy enough load, the manipulator could cause the mobile base to lose stability and possibly tip. In a situation like this, either a mechanism to increase stability needs to be put into place or more mass needs to be added to the mobile base.

- **Response Time:** A manipulator is usually much faster than its mobile base [4]. It also has less inertia, therefore it has a faster response time. This creates control and synchronization problems.

1.1.1 Current State of the Art

Due to their many benefits, mobile manipulators are starting to move from research labs into the industrial world. With new applications arising everyday, mobile manipulators will one day become common place. A few mobile-manipulators currently being used are shown here.

Currently, mobile manipulators are being used to do a variety of things. Figure 1.8 shows an industrial mobile manipulator used for charging and discharging large pieces

in forging, a very dangerous and labour intensive task. It has a total of 7-DOF. Its mobile base consists of two perpendicular tracks, one large track which travels in a straight line along the ground and a smaller track located on the mobile base itself. This gives the base 2-DOF. The manipulator accounts for the other 5-DOF as it is able to rotate about the vertical axis, move forward and backward, move up and down, close and open its grippers, and rotate its gripper about an axis parallel to its outstretched arm [5].



Figure 1.8: Mobile Manipulator used for Forging [5]

Figure 1.9 shows a prototype mobile manipulator that has been developed by the National Institute for Occupational Safety and Health. It has been specifically designed for use in mining with the base and manipulator being very maneuverable. However it can be used “for a variety of materials handling tasks in maintenance shops” [6] as well. It is capable of lifting up to 600 lbs. When the end-effector is attached to a heavy load, the operator guides the end-effector with his/her hand into the desired position and the end-effector remains there. It is used to replace a labour intensive activity and helps prevent workplace injuries.



Figure 1.9: Mobile Manipulator used for Lifting and Maneuvering Loads [6]

In Figure 1.10, a mobile manipulator used to remove coatings from “all but the largest commercial and transport aircraft” [7] is shown. It consists of a 6-DOF arm atop a 2-DOF mobile base. In order to remove coatings from an aircraft, the mobile base positions itself appropriately and then parks itself. The long 6-DOF arm with a specialized end-effector (for coating removal) is then utilized. The arm is able to swing left and right, pivot up and down, and extend. Its wrist contains the last 3-DOF and is able to pitch, roll, and yaw in order to conform to the contours of the aircraft’s body. It is a one of a kind mobile manipulator, being able to reach aircraft up to 42 feet off the ground [7].

Another similar mobile manipulator is one built for construction work, shown in Figure 1.11. It differs from the others as this mobile manipulator is mounted on a truck and has two smaller end-effectors atop a boom. Since the top of the boom is so heavy and so far above the base, the truck must secure itself with supporting legs. This specific mobile manipulator is being used to repair power lines.

A few companies sell small-scale industrial mobile manipulators. Figure 1.12 shows one of them, the mobile manipulator “MM-500” by Neobotix. This mobile manipu-



Figure 1.10: Mobile Manipulator for Removing Coating from Aircraft [7]

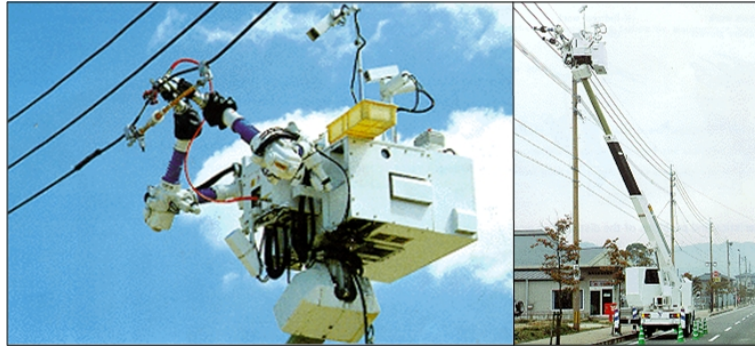


Figure 1.11: Mobile Manipulator used for Construction Work [8]

lator has a total of 9-DOF, 2-DOF from the mobile base and 7-DOF from the manipulator. The manipulator is capable of carrying a payload of 10 kg while the entire system has a payload of 40 kg. It can be used for a variety of material handling tasks. Mobile manipulators are most prevalent in law enforcement and the armed forces. They bring the advantages of a manipulator to the field and can be easily replaced, unlike their human counterparts. Figure 1.13 shows the tEODor mobile-manipulator system, where the EOD implies Explosive Ordnance Disposal. Manufactured by Kuchera Engineering, the tEODor system is composed of a 6-DOF arm mounted on a dual-tracked base. It is very maneuverable and robust, capable of going up slopes of 45° and operating in temperatures ranging from -20°C to $+60^\circ\text{C}$. The tEODor has a



Figure 1.12: Small-Scale Industrial Mobile Manipulator [9]

carrying capacity of 100 kg. Additionally, it has a tool magazine, allowing it to switch end-effectors on the go. It is highly flexible as well, with more than 40 additional tools that can be mounted to it for performing a variety of different tasks. For example, its end-effector can be mounted with a camera that relays information back to the operator sitting at its command station.



Figure 1.13: tEODor Mobile-Manipulator System [10]

Figure 1.14 shows the Packbot mobile-manipulator systems designed by iRobot. This mobile manipulator is highly customizable. Many different tools and end-effectors can be mounted on its dual-tracked mobile base, as shown in the figure. The Packbot also has an innovative feature in its two back tracks, called the dual QuickFlipTM track.

These tracks give it the ability to climb stairs and slopes up to 60° . The Packbot can travel at speeds up to 9.3 km/h. Like tEODor, it is very rugged. It can be thrown through a window and submerged under six feet of water and still work. They are controlled by a “game-style hand controller [11].” This makes it easy for the operator to learn how to use the system.



Figure 1.14: Packbots [11]

The Warrior mobile-manipulator system is shown in Figure 1.15. It too is designed by iRobot. It is very similar in design to the packbots with the dual QuickFlip™ track, being able to climb stairs. It can carry payloads of 150 lbs and is used for bomb defusal, surveillance, and route clearance.

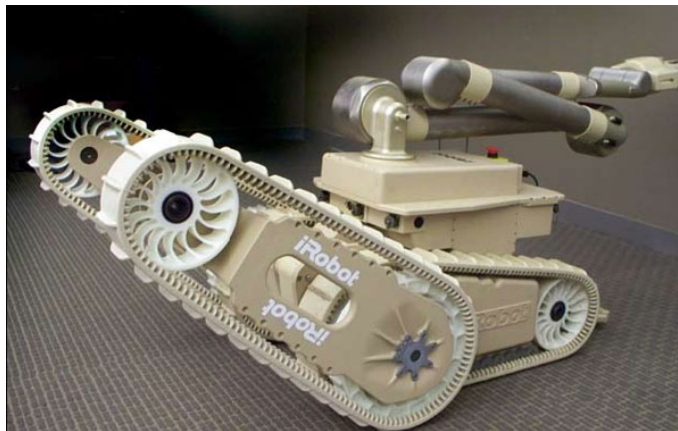


Figure 1.15: Warrior Mobile Manipulator [11]

Mobile manipulators do not have to be constrained to the terrestrial environment.



Figure 1.16: Shark Marine Sea-Dragon ROV [12]

A predominant majority of mobile-manipulators are ROVs (remotely operated underwater vehicles). Figures 1.16 and 1.17 show two such systems, the Shark Marine Sea-Dragon ROV and the Saab Seaeye Panther ROV. These types of mobile manipulators are comprised of underwater vessels with attached manipulators. They are able to travel thousands of metres to the bottom of oceans and lakes and perform delicate manipulation tasks required for such things as installing and repairing underwater structures, collecting underwater samples, and scavenging through ship wrecks.



Figure 1.17: Saab Seaeye Panther ROV [13]



Figure 1.18: Mobile Servicing System [14]

Mobile manipulators can also be found in space. The Mobile Servicing System (MSS) shown in Figure 1.18, is an example of one. The MSS is composed of three parts: the Canadarm2 (also known as the Space Station Remote Manipulator System (SSRMS)), the Mobile Base System (MBS), and the Special Purpose Dexterous Manipulator (Dextre). Canadarm2 is the manipulator and the MBS is the rail-guided mobile base. The MSS is located on the International Space Station (ISS) and was instrumental in its construction. The Canadarm2 has 7-DOF making it redundant. It can be affixed at various places on the ISS making it relocatable.

1.1.2 Literature Review

There has been significant work done in the field of mobile manipulators. Their many benefits have made them ideal for many industries. Their complex control has

made them ideal for study within academia. Work has been done on their design, motion control, and redundancy resolution. Their redundancy resolution methods are presented herein.

Work on redundancy resolution began with the study of redundant manipulators with fixed bases (i.e., manipulators fixed to the ground). One of the easiest ways to solve the redundancy problem of a manipulator is to fix or lock one of the DOF of the system (i.e., one of the joints) [15]. This is currently done in controlling the Canadarm2, however, this merely converts the redundancy problem into a non-redundant one and removes most, if not all, of the benefits of redundancy.

Most current control methods of manipulators and mobile manipulators rely on controlling the joint velocities. In order to understand these methods, an understanding of a few basic concepts is required. If $\mathbf{p}$ denotes the position and orientation of the end-effector and $\boldsymbol{\theta}$ denotes the joint angles of the manipulator, then the manipulator is redundant if $\dim(\mathbf{p}) \leq \dim(\boldsymbol{\theta})$. This means that more joint angles need to be specified than there are end-effector coordinates. In order to find $\mathbf{p}$ from $\boldsymbol{\theta}$, the forward displacement solution (FDS) is used. To find $\boldsymbol{\theta}$ from $\mathbf{p}$ the inverse displacement solution (IDS) is used.

Now if $\dot{\mathbf{p}}$ denotes the velocity of the end-effector and $\dot{\boldsymbol{\theta}}$ represents the rate of change of the joint angles, then the Jacobian, $\mathbf{J}$, represents the relationship between them:

$$\dot{\mathbf{p}} = \mathbf{J}\dot{\boldsymbol{\theta}} \quad (1.1)$$

In non-redundant manipulators, since $\dim(\mathbf{p}) = \dim(\boldsymbol{\theta})$, the Jacobian is square. By finding the inverse of the Jacobian, $\mathbf{J}^{-1}$, one can find $\dot{\boldsymbol{\theta}}$ from $\dot{\mathbf{p}}$:

$$\dot{\boldsymbol{\theta}} = \mathbf{J}^{-1}\dot{\mathbf{p}} \quad (1.2)$$

This gives the joint velocities based on the desired end-effector velocities. These joint

velocities are then sent to the controller.

If the Jacobian is non-invertible, i.e., $|\mathbf{J}| = 0$, it means that the manipulator is in a singularity, a configuration where it loses a DOF (e.g., when it is completely outstretched).

For redundant manipulators, the Jacobian is non-square. Therefore, it is not possible to take the inverse.

Many researchers have proposed methods around this. Whitney [15] proposed a solution to redundancy through the use of the pseudo-inverse of the Jacobian. This method constrains the redundancy so that minimum joint velocities of the manipulator are achieved. The pseudo-inverse is defined as:

$$\mathbf{J}^+ = (\mathbf{J} \times \mathbf{A}^{-1} \times \mathbf{J}^T)^{-1} \mathbf{J} \times \mathbf{A}^{-1} \quad (1.3)$$

where $\mathbf{A}$ is a matrix that the user adjusts based on the desired movement.

This method can be used to achieve a variety of objectives. Liegeois [16] used this method to optimize a 6-DOF for joint range availability. As well, Klein [17] and Maciejewski and Klein [18] used this method for obstacle avoidance.

This method does, however, pose a problem. The use of the pseudo-inverse is unstable when the manipulator is in a singular position. In addition, it does not allow for “control of inner parts of the structure” [19]. This means that the configuration of the joints cannot be specified using this method and the joint angles are a product of the pseudo-inversion.

Baillieul [20] presented the extended Jacobian technique. In this method, a task is defined such as optimizing the configuration for manipulability. This condition is then added to the Jacobian in order to make it square. With the Jacobian square, it can be inverted and the joint speeds can be found, making it computationally easier to solve. As well, the solution is cyclic, meaning that if the end-effector is specified to go in a path where the start point is the same as the end point, the configuration of

the robot would be the same at the start and at the end [3].

Egeland [21] and Sciavicco and Siciliano [22] both solved the redundancy by proposing the use of an augmented task space. This method involves setting up n additional task constraints where n is the extra DOF of the system. The constraint tasks are defined with respect to the joint velocities, $\dot{\theta}$. This then allows the constraints to be placed directly into the Jacobian, making it square and easily invertible.

The problem with the extended Jacobian method and the augmented task space is that they are prone to algorithmic singularities [20]. Algorithmic singularities occur when either the extended Jacobian or the augmented Jacobian become singular yet the manipulator is not in an actual physical singularity.

These methods for redundant manipulators can easily be applied to mobile manipulators, as they are essentially one and the same, with the exception of the nonholonomic constraint of the mobile base. The nonholonomic constraint defines how a mobile base can travel. Due to the friction of the wheels, the mobile base is limited to a set of paths that it can take. It cannot for example move directly sideways. In order to do so, it must first rotate 90° and then move forwards.

Tchon and Malek [23] developed a singularity robust Jacobian inverse specifically aimed at mobile manipulators that incorporates this nonholonomic constraint. They have simulated their algorithm with success.

Seraji [24, 25] proposed a method similar to that of the task space augmentation proposed in [21] and [22]. In [24] this method is applied to holonomic mobile manipulators, while in [25] it is extended to nonholonomic mobile manipulators. In this approach, the mobile manipulator is modeled as one system. The end-effector velocity is modeled using the joint velocities and the mobile base's wheel velocities, since velocity control is used. To resolve the Jacobian, a set of additional task constraints are added to the Jacobian. For nonholonomic mobile manipulators, a subset of the additional task constraints must be the nonholonomic constraints. The number of

additional tasks must be equal to or greater than the degree of redundancy. If the number of additional tasks is equal to the degree of redundancy, then the regular Jacobian inverse is taken. If, however, the number of additional tasks is greater than the degree of redundancy, the use of a pseudo-inverse is required. This approach is both very simple and computationally efficient, as simulated results have proven. The additional task constraints are, however, functions of time, meaning that the desired trajectory of the end-effector should be known before hand. This is not possible when an operator is directly tele-operating the system.

Fourquet and Renaud [26] used a similar method. They proposed the use of a set of additional tasks as well as the minimization of a quadratic criterion to control a nonholonomic mobile manipulator. This method is based on a predefined trajectory of the end-effector and the motion of the manipulator and mobile-base is coupled at all times.

In order to solve the inverse kinematic problem for the control of nonholonomic mobile manipulators, Wang and Kumar [27] specified a compliance function for each joint which provided control on a local level, while optimizing additional criteria to provide control on a global level. The algorithm was simulated and required a predefined end-effector trajectory.

Others have focused primarily on singularities. Tan et al. [28] proposed a method that explicitly identifies the singular configurations of the manipulator and tries to avoid them in order to control a mobile manipulator. This control method identifies i conditions for singularity. It sets each condition to γ_i which then acts as a measure of how far the manipulator is from a singularity, with $\gamma_i = 0$ occurring at a singular configuration. Therefore, the closer γ_i is to 0, the closer the manipulator is to a singularity. By monitoring γ_i , therefore, and making sure that it is greater than a specified minimum value, the manipulator's singularities are avoided. This system is specifically designed for force/position control.

Others have taken the approach of only operating one of the two systems (either the manipulator or mobile base) at a time. Takubo et al. [29] developed a control scheme for a mobile manipulator system through the use of a virtual impedance wall. In this scheme, the workspace of the manipulator is predefined as to avoid singularities, joint limits, and contact with the mobile base. The mobile-manipulator is used as a mobile-assist system (i.e., to help the operator carry an object). The operator’s movement of the object dictates the movement of the manipulator. Only the manipulator moves when its end-effector is within the preferred region. As the manipulator hits the end of this preferred region (the impedance wall), “the mobile base moves by a repulsive force from the impedance wall” [29]. Thereby, the mobile-base only moves when it has to.

Anderson et al. [30] also took this approach. They briefly outline the automatic workspace extension control scheme. An operator controls a 3-DOF manipulator mounted on a nonholonomic base. As the manipulator reaches outside of its predefined workspace, the mobile base moves forward to compensate. This scheme only considers the forward direction however, and lateral directions are ignored due to the nonholonomic constraints. The conditions on how the mobile-base stops moving are not specified.

Shin et al. [31] implemented a system where an end-effector trajectory is specified and the mobile base only moves once the trajectory of the manipulator is out of reach. Once the mobile base reaches its directed pose, the manipulator continues to follow the trajectory. This process is repeated over and over.

As mentioned before, a mobile manipulator’s position may be optimized in order to complete a desired task. However, problems are encountered when there is a transition between tasks. For example, in order to reach an object, it is optimal for the manipulator to be outstretched. However, once the manipulator is outstretched, this configuration may not be optimal to pick up the object. Pin and Culioli [32] discuss a

methodology that involves multi-criteria optimization that solves this problem. This involves the need to “forecast” the future task and optimize the current and future task at the same time in order to achieve the transition.

Work on neural network control of mobile manipulators has also been conducted. A neural network is “an interconnected assembly of simple processing elements, *units* or *nodes*, whose functionality is loosely based on the animal neuron” [33]. These neural networks are able to be taught and can learn different behaviours. Gao et al. [34] proposed a controller based on neural networks and implemented it on a mobile manipulator consisting of a 5-DOF arm. Lin and Goldenberg [35] proposed and implemented the use of two neural network controllers to control the manipulator and mobile base separately. Chen and Zalzal [36] implemented a neural network controller on a mobile manipulator. The neural network controller was able “to learn on-line and produce compensation torques to reduce the tracking errors [36].” It was also trained to deal with wheel slippage that violated the nonholonomic constraints. A type of mobile manipulator is an underwater vehicle manipulator system (UVMS) (ROVs with robot arms). These systems differ from land based mobile manipulator systems as they have more DOF. Antonelli and Chiaverini [37] proposed control schemes for UVMS based on a task priority system. Accurate control of the manipulator is the primary task. A secondary task such as reducing the UVMS energy consumption or increasing its dexterity is also implemented. If both tasks cannot be achieved at the same time, then the proposed control scheme only implements the primary task. The authors state that the method is robust against the occurrence of algorithmic singularities [37]. In [38], the same authors integrated this task priority system with fuzzy logic, allowing control of multiple secondary tasks.

1.1.3 Problem Statement and Goal

Many problems exist with the above mentioned methods. For one, some of these methods, do not avoid singularities (i.e., they are not singularity robust). They allow the manipulator to lose a degree-of-freedom, which in turn causes them to be less dexterous. If a mobile-manipulator is in a singularity, its performance is reduced, and not utilizing its full potential. Additionally, many of these techniques produce algorithmic singularities. These are not actual physical singularities, but ones that come about due to manipulation of the Jacobian matrices. This is a result of having modeled the mobile manipulator as one body, rather than two separate entities.

Most, if not all, of the research focuses on solving the kinematics of the system. In testing the algorithms, predefined end-effector trajectories are input into the system. Little attention is paid to how the system will be operated or tele-operated. No mention is made of how the operator will input the desired motion and no consideration is given to the fact that the operator will likely not know the end-effector trajectory before hand. The extension between theoretical control and actual control is not made. Furthermore, testing is mostly done using simulations. Few researchers actually implement their control algorithms on physical testbeds.

In actuality, most mobile-manipulator systems that have been built commercially are controlled by two joysticks, one joystick for the manipulator and one for the mobile base as shown in Figure 1.19. This makes it very difficult to control mobile manipulators. It takes a lot of skill, concentration, and dexterity as the task occupies both hands, and can be mentally draining. Controlling a non-redundant system is easy as there is only a finite number of ways to accomplish a task. However, with redundancy the user has to consider things he/she normally would not. Furthermore, the greater the redundancy (i.e., the more extra DOF), the more difficult the system is to control. An underwater ROV can have up to 12-DOF. Manually controlling a underwater mobile manipulator “is a long and stressing task” [39]. The operating

time and the quality of work “are limited due to operator fatigue” [39].

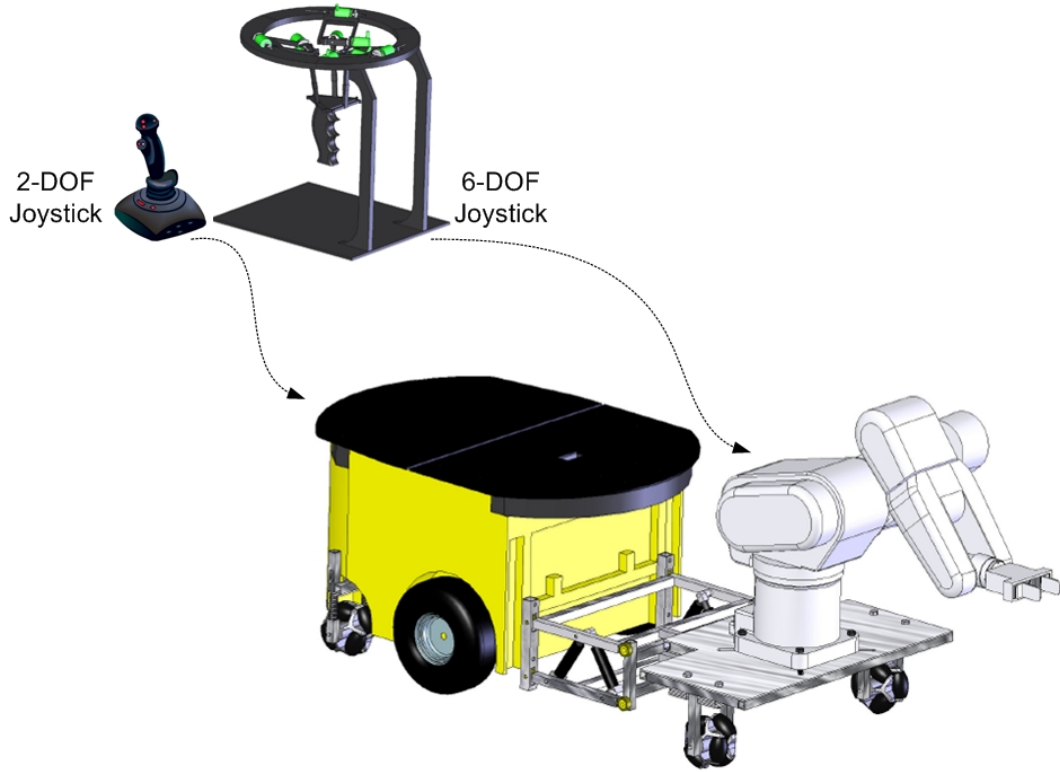


Figure 1.19: Traditional Control of Mobile Manipulators

In addition to controlling the mobile manipulator, the operator may also need to check the control panel, document data, communicate with other team members, and even possibly control a second robotic arm. In order to do an effective job, the operator should be focused on controlling the desired motion of the end-effector. The more complicated the control is and the more stress it involves, the higher the room for error. This is an example of where technology has allowed individuals to do things they have never been able to do before, but placed a heavy burden on their skills. Ideally, technology should reduce the required skill of such a task and make it easier for the operator. Hence the inspiration for this thesis. By designing an intuitive controller, the operator will be able to do a better job, in a more efficient manner.

A control algorithm that avoids the above problems is proposed. The control algorithm avoids singularities, both physical and algorithmic from the way that it is

formulated. It takes advantage of the redundancy, placing the manipulator into a preferable configuration when needed. It relies on one input device, rather than two (or more), making it much simpler to use. It provides unified, intuitive, and coordinated control. This means that it has a *sense* of the operator's intentions. Most importantly it is simple to use; little training is required in order to learn how to use it. As well, it takes the operator into consideration rather than just treating the problem theoretically.

The goal of this research is the creation of a novel control algorithm that can be adopted for the control of a host of mobile-manipulator systems. It can be used for completing tasks in the fields of underground mining, nuclear fuel and hazardous material handling, bomb defusal, underwater construction/repair, and industrial work. It is desired that this algorithm will form the basis for expansion into systems with high degrees of redundancy, such as ROVs.

1.2 Scope

The scope of this thesis involves the following:

1. Design and implementation of a novel algorithm for the control of mobile-manipulator systems. The control algorithm must be both feasible and intuitive in nature. It must also overcome the current problems with most mobile-manipulator control algorithms.
2. Development of a testbed (both the hardware and software) so that implementation and testing of the control algorithm is possible.
3. Implementation of the algorithm on the prototype testbed in order to show the algorithm's effectiveness.

Due to the large scope of such a project, some assumptions are made. Obstacle avoidance will not be dealt with. Such a topic would be worthy of a master's thesis of its own. Additionally, it will be assumed that the mobile manipulator travels on perfectly flat surfaces only. The dynamic interaction between the manipulator and mobile base will also be ignored. It will be assumed that both systems are stable and that there is little vibration.

1.3 Thesis Outline

This thesis is organized as follows. Chapter 2 is dedicated to the hardware of the mobile-manipulator system testbed. It discusses the manipulator, mobile base, joystick, and the way the system is physically interconnected.

In Chapter 3, mathematical models of the manipulator and mobile base are derived. This includes the forward and inverse displacement solutions as well as the manipulator's singularities. It also defines the equations of the nonholonomic constraint of the mobile base.

Chapter 4 discusses the novel control algorithm that has been developed. The implementation of the control algorithm is thoroughly discussed along with the software and the system architecture in Chapter 5.

Chapter 6 discusses the results of the control algorithm implementation, its effectiveness, as well as problems encountered and what needs to be done to fix these problems.

Finally, in Chapter 7, conclusions and future work are discussed.

Chapter 2

Hardware

A mobile manipulator, dubbed *Jasper*, has been constructed over the last few years in the Mechatronic and Robotic Systems Laboratory, UOIT. Jasper, shown in Figure 2.1, consists of a manipulator mounted to the front of a mobile robotic base through the use of a parallelogram connecting device. This configuration allows Jasper to be able to pick up and reach items located on the ground. It also lets the manipulator, in effect, *lead* the mobile base. The parallelogram connecting device incorporates a 2-DOF (degrees-of-freedom) passive suspension into the mobile manipulator, allowing it to traverse over uneven terrain with all of its six wheels on the ground at all times. The device dampens any pitch and roll between the two systems as well as any vibrations. Supporting most of the mass of the manipulator are two sets of omnidirectional wheels. These wheels are passive and can spin both forward and sideways as they are composed of curved circular barrels. They work in the same way as caster wheels, however, they do not lock up. The complete details of the mechanical design of Jasper are documented in [40].

Jasper has a total of 8-DOF, 2-DOF from the mobile base and 6-DOF from the arm. It is to be used as the platform for testing, debugging, and refining control algorithms for mobile manipulators. More details in regards to the individual systems are provided

below.

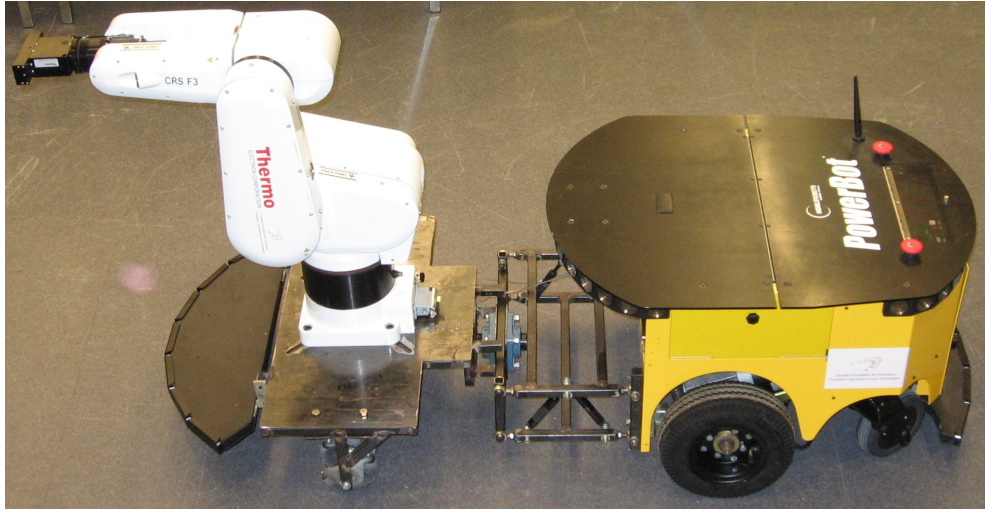


Figure 2.1: Jasper Mobile-Manipulator System

2.1 Manipulator

The manipulator of Jasper is a CRS F3 articulated arm produced by Thermo Fisher Scientific (formerly Thermo Electron Corporation). It has 6-DOF, each of which is attributed to its six joints shown in Figure 2.2. It has a classic robot arm configuration in which the first three joints are used to control the position of the wrist centre, while the last three joints form a spherical group and are used to control the orientation of the end-effector.

The arm's specifications are listed in Table 2.1.

The manipulator is connected to a CRS C500C Controller also produced by Thermo Fisher Scientific. Its specifications are listed in Table 2.2.

A cable from the controller connects to the manipulator, both powering it as well as transferring movement instructions to it. The manipulator is programmed through the use of the RAPL-3 robot programming language, which is very similar to the C programming language. For more details regarding the CRS F3 manipulator and CRS C500C Controller see [41].

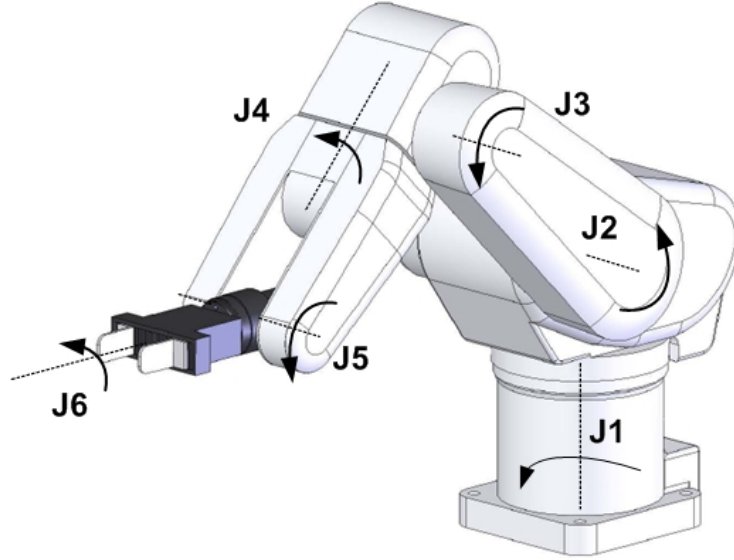


Figure 2.2: Joints of the CRS F3 Articulated Arm

Table 2.1: Thermo Fisher Scientific CRS F3 Arm Specifications [41]

Weight	52 kg (115 lb)
Nominal Payload	3 kg (6.6 lb)
Reach	710 mm (28 in.)
Repeatability	± 0.05 mm (0.002 in.)
Encoder Resolution	2048 counts/motor turn
Maximum Linear Speed	4 m/s
Drive System	Electromechanical, brushless motors
Transmission	Harmonic Drives

Mathematical models of the kinematics of the manipulator are developed and shown in Chapter 3.

2.2 Mobile Base

The mobile base is a PowerBotTM AGV (automated guided vehicle) produced by ActivMedia Robotics, shown in Figure 2.3. It was designed for “autonomous, intelligent delivery and handling of large payloads” [42]. It consists of four wheels, two large driven wheels at the front and two smaller passive caster wheels at the back. It uses

Table 2.2: Thermo Fisher Scientific CRS C500C Controller Specifications [41]

Dual Microprocessor Design	133 MHz i486DX (system processor) 60 MHz TMS320C31 DSP (motion control)
Memory	4 MB RAM user memory 512KB NVRAM for application storage 1 MB flash memory for system firmware
System Connections	2 standard serial I/O ports 1 console serial port

differential steering, whereby one wheel turns faster than the other in order to turn. As such, it has 2-DOF being able to rotate about the vertical axis and move forward. Hence, its motion spans a horizontal plane.



Figure 2.3: PowerBot™ AGV [43]

The PowerBot™ AGV has 14 sonar sensors at the front and 14 sonar sensors at the back. These allow the PowerBot to avoid obstacles or do such things as map a room. For the use of this project they are not necessary and have been disabled. The PowerBot also has bump sensors located on the front and back. These serve as a last measure of safety, in case the sonar fails. Since the parallelogram connecting device is connected to the front of the PowerBot, the front bump sensors have been

moved from their original position to the front of the connecting device (see Figure 2.1). These sensors stop the PowerBotTM AGV from moving once pressed.

The PowerBot’s specifications are listed in Table 2.3.

The PowerBot is programmed using “a C++ based open-source development environment,” [42] called ActivMedia Robotics Interface for Applications (Aria). This programming library contains all the necessary functions to acquire sensor data and control the PowerBot. For more details regarding the PowerBotTM AGV see [42].

Mathematical models of the mobile base are derived and presented in Chapter 3.

2.3 Joystick

A 6-DOF joystick (“6-DOF Master-Controller”), originally built by RSI Research Ltd. is shown in Figure 2.4. It is to be used as the input device to control Jasper. The joystick is based on a three-branch parallel mechanism. Each of the branches contains three links. Figure 2.5 shows how the links are connected and how they rotate about one another. Link 1 is connected to the circular metallic frame of the joystick. It is able to rotate with respect to it. Link 2 rotates about the end of Link 1 with an axis of rotation parallel to the axis of rotation of Link 1 about the metal frame. Link 3 is connected to Link 2 and connects to the edge of the triangular end-effector platform with a spherical wrist. All three branches are joined together at the end-effector platform. At the bottom of the triangular base is an ergonomic handle, which the user controls. The joystick can move forward and backward, up and down, left and right, as well as pitch, roll, and yaw.

There are three digital encoders on each branch, nine in total. Only six encoders are necessary to compute the joystick’s position and orientation. The three additional encoders are “exploited to allow self-calibration and fault-tolerant operation” [45]. Fault-tolerant operation means that if one of the encoders should fail, the position

Table 2.3: PowerBot™ AGV's Specifications [44]

PHYSICAL DIMENSIONS	
Size ($l \times w \times h$)	85cm $\times$ 63cm $\times$ 47cm
Mass (with minimum battery capacity)	120 kg
Payload of platform with included battery	100kg flat 60 kg 8% grade
Drive Wheel Diameter	27 cm
Drive Wheel Width	9 cm
Caster Wheel Diameter	19cm
POWER	
Battery	2 $\times$ 12V sealed, lead acid
Charge	2160 watt-hr
Continuous Run-Time	4.5 hrs
Recharge time	2.5 hrs
DRIVE	
Maximum Speed	2.1m/sec
Traversable slope max	15% grade
Drive Wheel Diameter	27 cm
Drive Wheel Width	9 cm
Turn Radius	0
Gear ration	22.3:1
Encoders	1000 counts/rotation
ONBOARD COMPUTER	
Processor Type	Intel®Pentium®4 CPU
Processor Speed	2.40 GHz
Processor Cache	512 KB
Hard Drive Size	38.5 GB
RAM	500 MB
Communications	4 RS-232 Serial Ports and Wireless Adapter
Operating System	Red Hat Linux release 7.3
CONTROLLER	
Processor Type	Hitachi H8S
Flash Memory	1 MB

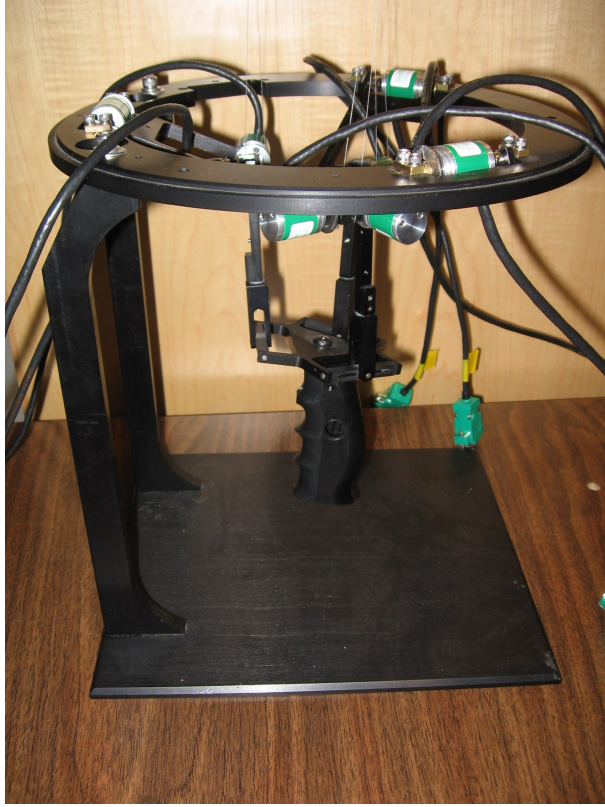


Figure 2.4: Six Degree-of-Freedom Spatial Joystick

and orientation of the joystick can still be calculated. Up to two encoders can fail with the current configuration.

The joystick has been digitized to improve accuracy as well as reduce noise with the original analog potentiometers having been replaced with digital encoders (Gurley Precision Instruments R120 Rotary Incremental Encoders). The joystick's position is accurate to about $\pm 0.25 \text{ mm}$ [45]. The digital encoders are incremental, hence calibration is required every time the joystick is first powered up. Calibration involves moving the joystick around in a random fashion for a short period of time. The collected data, along with the known link lengths and knowledge of the inverse kinematic solution of the joystick, is then used to determine encoder offsets necessary to find the actual position and orientation. Specific details in regards to calibration and the solution to the inverse kinematics of the 6-DOF joystick can be found in [45].

All nine encoders are connected to a conversion box through serial cables, see Figure

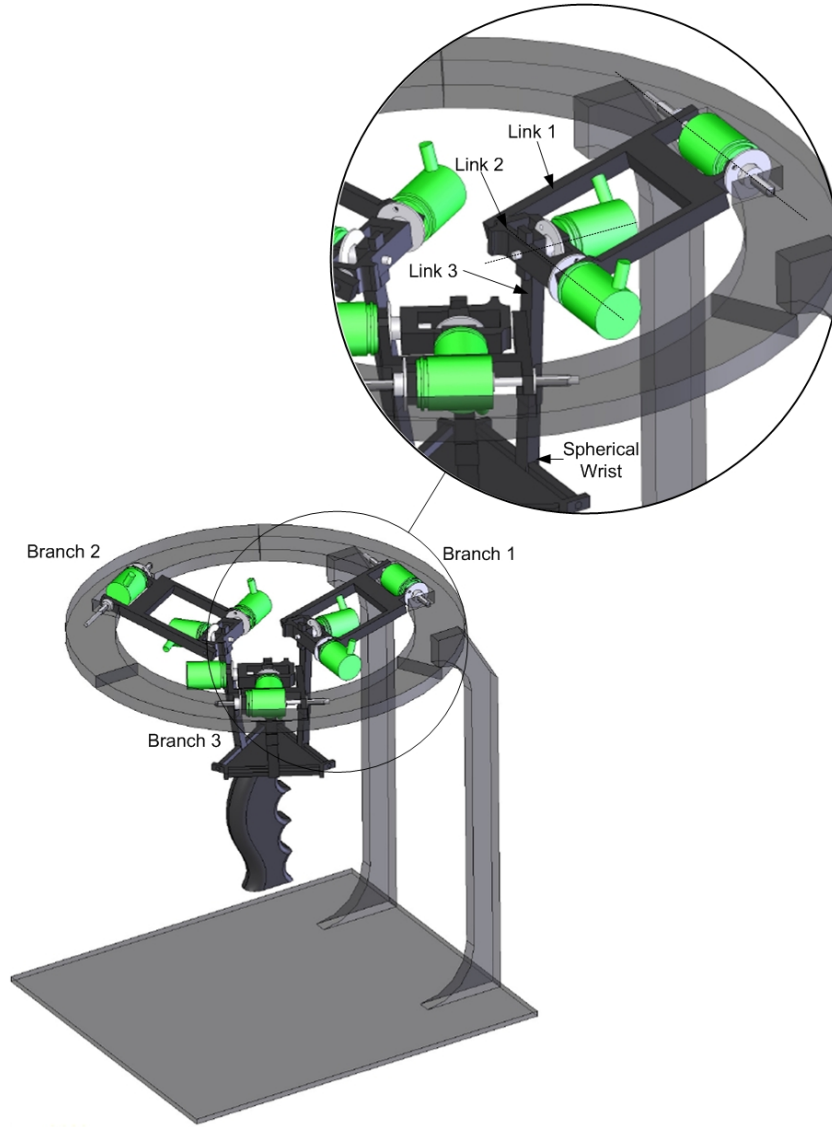


Figure 2.5: Six Degree-of-Freedom Spatial Joystick

2.6. This conversion box houses two 50-pin ribbon cable terminal blocks (mounted on a DIN rail). This routes the data flowing through the serial cables into two 50-pin ribbon cables. The two 50-pin ribbon cables are fed into the RT-LAB computer which runs QNX, a real-time operating system (see Figure 2.7).

Each of the 50-pin ribbon cables is connected to one of two Sensoray 626 input/output cards. The 50-pin ribbon cable is split into two 25-pin ribbon cables (two per card). These 25-pin ribbon cables are each connected to a 26-pin IDC ribbon connector. This means that one of the pins of the ribbon cable is not connected per input/output card.

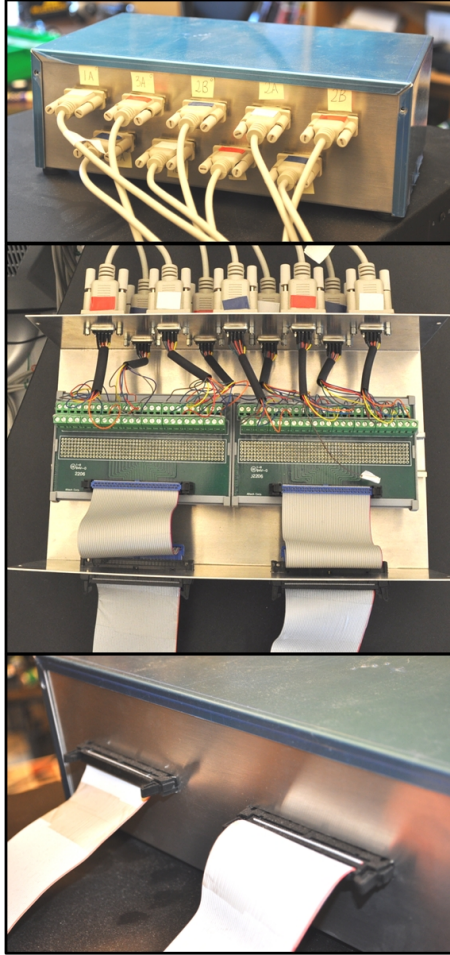


Figure 2.6: Conversion Box

This is believed to be a manufacturer’s error. It prevents one of the index pins from the encoder from being connected. In order to provide uniformity across all encoders, no index pins from the encoders were connected at all. This prevented hardware indexing and hence, software indexing was used instead.

2.4 Operations Computer

The Operations Computer is an RT-LAB Engineering Simulator designed by Opal-RT. Conceptually, it is at the center of the system. This computer runs on the QNX Neutrino real-time operating system (RTOS) v6.3.2. QNX has been specifically designed for real-time tasks such as the one described herein and is known for its reliability

and speed. This computer is where control decisions are made. The specifications for the Operations Computer are shown in Table 2.4.

Table 2.4: Operations Computer Specifications [41]

Processor Type	AMD Athlon™64 3500+
Processor Speed	2210 MHz
RAM	512 MB
Hard Drive	40 GB
Operating System	QNX Neutrino v6.3.2

2.5 Overall System

The overall system is shown in Figure 2.7. The Console Computer is connected to the C500C Controller through the use of a serial cable. The Console Computer is used to compile and upload programs to the C500C robot arm controller. It also acts as a console allowing the user to start and terminate programs on the controller and allows graphic output from the C500C controller to be displayed, so the user can see data from the manipulator.

The Operations Computer is the “heart” of the system. It is connected to the joystick (via the conversion box). The Operations Computer is also connected to the C500C Controller through the use of a null modem cable. This allows the programs on the Operations Computer to communicate with those programs directing the manipulator. Lastly, the Operations Computer is connected to the PowerBot through a wireless Ethernet connection. This allows the programs on the Operations Computer to access those running on the mobile base.

It should be noted that in an ideal situation the manipulator controller would be onboard the mobile manipulator. In this case it was not possible as the controller is very large and requires a relatively large power supply.

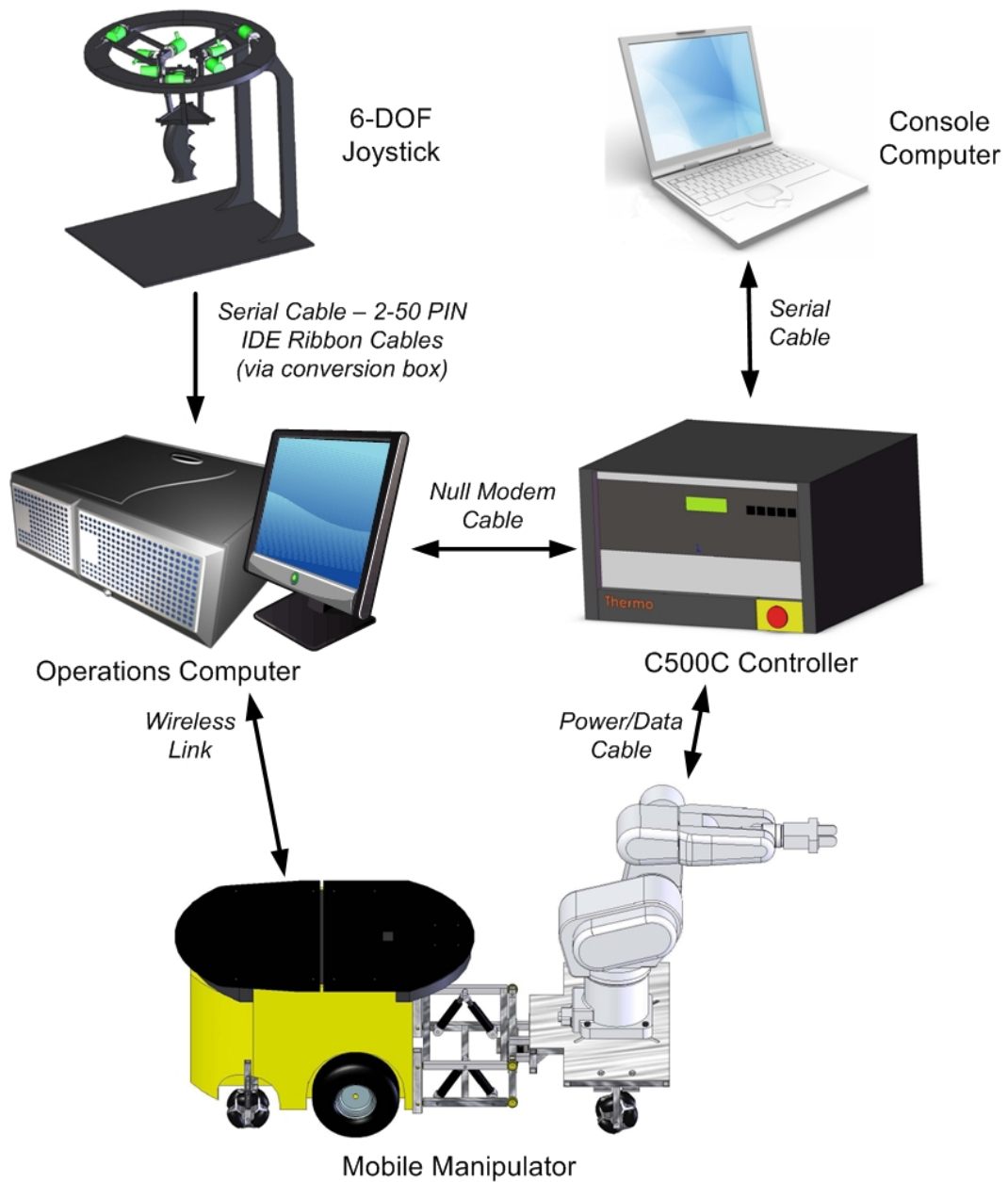


Figure 2.7: Overall System Architecture

Chapter 3

Mobile-Manipulator Kinematics

Before the control algorithm is discussed and implemented, it is important to have accurate mathematical descriptions of the system so that it can be modeled and controlled properly. In this chapter, mathematical models of both the CRS F3 articulated arm and the PowerBotTM AGV are developed. In the case of the manipulator, the forward displacement solution, inverse displacement solution, and Jacobian are found. The Jacobian is then used to find the singularities of the manipulator and these are analyzed. The model of the mobile base describes its position based on its left and right wheel speeds.

Many times mobile manipulators are modeled as one entity: manipulator connected to base. However, this makes the model extremely complicated. As a result, the mobile manipulator will be modeled as two separate entities: manipulator and mobile base. This will make the understanding the model much clearer and make the control much simpler.

3.1 Manipulator

The CRS F3 articulated arm has a very simple configuration (see Figure 2.2). Joints 1, 2, and 3 are of the $R \perp R \parallel R$ configuration where R denotes a revolute joint, $\perp$

denotes two joint axes being perpendicular to one another, and $\parallel$ denotes two joint axes being parallel to one another. They are used in positioning the end-effector. Joints 4, 5, and 6 are of the $R \perp R \perp R$ configuration. They form what is called a spherical wrist or spherical group. Hence, they are used in orienting the end-effector. This type of decomposition further simplifies the control.

3.1.1 Zero Displacement Diagram

A zero displacement diagram of the F3 articulated arm is shown in Figure 3.1. Zero-displacement diagrams illustrate how the frames of the joints are related to one another.

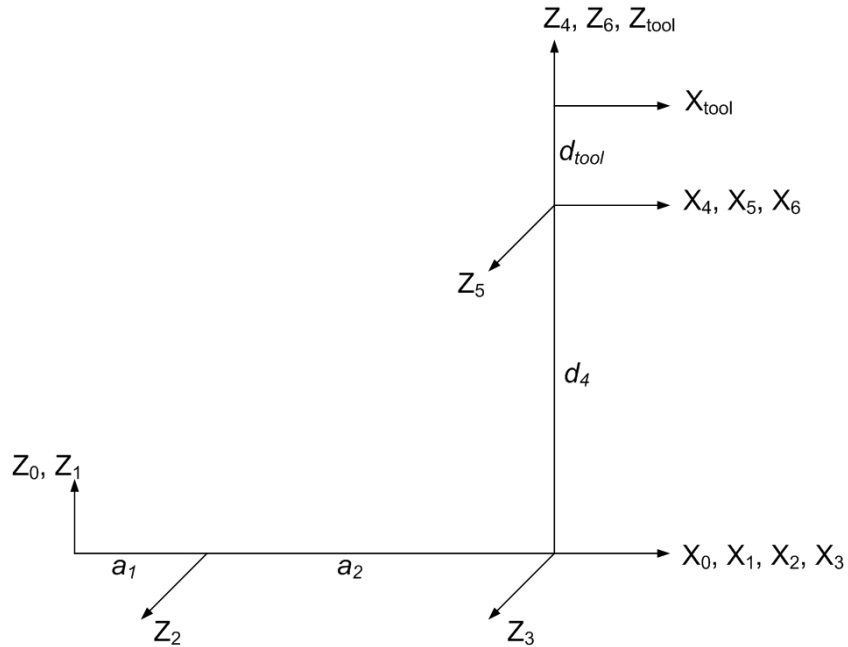


Figure 3.1: Zero Displacement Diagram of the F3 Manipulator

With respect to revolute joints, the Z_i axis indicates the axis of revolution of the i th joint. With respect to prismatic joints, it indicates the axes of translation of the i th joint. The x_i axis is the axis of the i th frame that is perpendicular to z_i and z_{i+1} .

The zero displacement diagram outlines the robot arm's configuration and allows for the derivation of the Denavit and Hartenberg (D&H) parameters, described in the

next section.

3.1.2 Denavit and Hartenberg Parameters

Robotic links can be described using Denavit and Hartenberg (D&H) parameters $(\alpha_{i-1}, a_{i-1}, d_i, \theta_i)$ [1] where α_{i-1} represents the angle between z_{i-1} and z_i using right hand convention, a_{i-1} represents the distance between z_{i-1} and z_i , d_i represents the distance between x_{i-1} and x_i , and θ_i represents the angle between x_{i-1} and x_i . These parameters describe the relationships between the joints and the links and can mathematically describe the configuration of an entire manipulator through:

$${}^{i-1}_i\mathbf{T} = \mathbf{R}_x(\alpha_{i-1}) \mathbf{D}_x(a_{i-1}) \mathbf{R}_z(\theta_i) \mathbf{D}_z(d_i) \quad (3.1)$$

where ${}^{i-1}_i\mathbf{T}$ is a homogeneous transform describing frame F_i with respect to frame F_{i-1} , $\mathbf{R}_x(\alpha_{i-1})$ represents a rotation about the x axis by α_{i-1} , $\mathbf{D}_x(a_{i-1})$ represents a translation along the x axis by a_{i-1} , $\mathbf{R}_z(\theta_i)$ is a rotation about the z axis by θ_i , and $\mathbf{D}_z(d_i)$ represents a translation along the z axis by d_i .

The homogeneous transformation matrix is found as:

$${}^{i-1}_i\mathbf{T} = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.2)$$

where c denotes cosine and s denotes sine.

The D&H Parameters for the CRS F3 are shown in Table 3.1. They summarize the zero displacement diagram and are used to construct the forward displacement solution (FDS).

Table 3.1: Denavit and Hartenberg Parameters for the CRS F3 Manipulator

$i - 1$	α_{i-1}	a_{i-1}	d_i	θ_i	i
0	0	0	0	θ_1	1
1	$\frac{\pi}{2}$	a_1	0	θ_2	2
2	0	a_2	0	θ_3	3
3	$-\frac{\pi}{2}$	0	d_4	θ_4	4
4	$\frac{\pi}{2}$	0	0	θ_5	5
5	$-\frac{\pi}{2}$	0	0	θ_6	6
6	0	0	d_{tool}	0	$tool$

3.1.3 Forward Displacement Solution of the Wrist Centre

The forward displacement solution (FDS) of a manipulator calculates the position, ${}^M\mathbf{p}_{wc}$, and orientation, $[\mathbf{n} \ \mathbf{o} \ \mathbf{a}]$, of the manipulator's wrist centre based on the joint angles (θ_1 to θ_6). It is represented by the homogeneous transform describing the tool frame, F_{tool} , with respect to the base frame, F_0 :

$${}_{tool}^0\mathbf{T} = {}_1^0\mathbf{T}_2^1\mathbf{T}_3^2\mathbf{T}_4^3\mathbf{T}_5^4\mathbf{T}_6^5\mathbf{T}_{tool}^6\mathbf{T} \quad (3.3)$$

where ${}_i^{i-1}\mathbf{T}$ is defined in equation (3.2).

Note that:

$$\begin{aligned} {}_6^0\mathbf{T} &= {}_{tool}^0\mathbf{T} {}_{tool}^6\mathbf{T} = {}_{tool}^0\mathbf{T} ({}_{tool}^6\mathbf{T})^{-1} = \begin{bmatrix} \mathbf{n} & \mathbf{o} & \mathbf{a} & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (3.4)$$

where

$$\begin{aligned}
\mathbf{n} &= \begin{bmatrix} ((c_1 c_{23} c_4 - s_1 s_4) c_5 - c_1 s_{23} s_5) c_6 - (c_1 c_{23} s_4 + s_1 c_4) s_6 \\ ((s_1 c_{23} c_4 + c_1 s_4) c_5 - s_1 s_{23} s_5) c_6 - (s_1 c_{23} s_4 - c_1 c_4) s_6 \\ (s_{23} c_4 c_5 + c_{23} s_5) c_6 - s_{23} s_4 s_6 \end{bmatrix} \\
\mathbf{o} &= \begin{bmatrix} -((c_1 c_{23} c_4 - s_1 s_4) c_5 - c_1 s_{23} s_5) s_6 - (c_1 c_{23} s_4 + s_1 c_4) c_6 \\ -((s_1 c_{23} c_4 + c_1 s_4) c_5 - s_1 s_{23} s_5) s_6 - (s_1 c_{23} s_4 - c_1 c_4) c_6 \\ -(s_{23} c_4 c_5 + c_{23} s_5) s_6 - s_{23} s_4 c_6 \end{bmatrix} \\
\mathbf{a} &= \begin{bmatrix} -(c_1 c_{23} c_4 - s_1 s_4) s_5 - c_1 s_{23} c_5 \\ -(s_1 c_{23} c_4 + c_1 s_4) s_5 - s_1 s_{23} c_5 \\ -s_{23} c_4 s_5 + c_{23} c_5 \end{bmatrix} \\
\mathbf{p} &= \begin{bmatrix} -c_1 s_{23} d_4 + c_1 c_2 a_2 + c_1 a_1 \\ -s_1 s_{23} d_4 + s_1 c_2 a_2 + s_1 a_1 \\ c_{23} d_4 + s_2 a_2 \end{bmatrix}
\end{aligned}$$

and where c_{ij} and s_{ij} denote $\cos(\theta_i + \theta_j)$ and $\sin(\theta_i + \theta_j)$, respectively.

3.1.4 Inverse Displacement Solution of the Wrist Centre

The inverse displacement solution (IDS) is used to calculate the required joint angles based on the position and orientation of the wrist center. It is the opposite of the forward displacement solution. In the inverse displacement solution, it is possible for an angle to have more than one solution, therefore, the number of solutions for each joint angle are noted.

Given the desired pose of the wrist, the solution for θ_1 can be found as:

$$\theta_1 = \text{atan2}(p_y, p_x) \quad (3.5)$$

$$\theta_1 = \text{atan2}(-p_y, -p_x) \quad (3.6)$$

where atan2 denotes a quadrant corrected arctangent function. Note that there are two possible solutions for θ_1 .

The solution for θ_2 can be found as:

$$\theta_2 = \text{atan2}(k, j) \pm \text{atan2}\left(\sqrt{j^2 + k^2 - l^2}, l\right) \quad (3.7)$$

where

$$j = 2a_1a_2 - 2a_2p_xc_1 - 2a_2p_ys_1$$

$$k = -2a_2p_z$$

$$l = d_4^2 - p_x^2 - p_y^2 - p_z^2 - a_1^2 - a_2^2 + 2a_1p_ys_1 + 2a_1p_xc_1$$

Note there are two possible solutions for θ_2 .

The solution for θ_3 can be found as:

$$\theta_3 = \text{atan2}(m, n) - \theta_2 \quad (3.8)$$

where

$$m = \frac{c_1p_x + s_1p_y - a_3c_2 - a_2}{-d_4}$$

$$n = \frac{p_z - a_3s_2}{d_4}$$

With θ_1 to θ_3 determined, the solutions for θ_4 to θ_6 can be determined. For θ_4 :

$$\theta_4 = \text{atan2}(q, t) \quad (3.9)$$

where

$$q = \frac{s_1 r_{13} - c_1 r_{23}}{s_5}$$

$$t = \frac{-c_1 c_{23} r_{13} - s_1 c_{23} r_{23} - s_{23} r_{33}}{s_5}$$

For θ_5 :

$$\theta_5 = \text{atan2}(u, v) \quad (3.10)$$

where

$$u = -r_{13}c_1c_{23}c_4 + r_{13}s_1s_4 - r_{23}s_1c_{23}c_4 - r_{23}c_1s_4 - s_{23}c_4r_{33}$$

$$v = -c_1s_{23}r_{13} - s_1s_{23}r_{23} + c_{23}r_{33}$$

For θ_6 :

$$\theta_6 = \text{atan2}(x, y) \quad (3.11)$$

where

$$x = -r_{11}c_1c_{23}s_4 - r_{11}s_1c_4 - r_{21}s_1c_{23}s_4 + r_{21}c_1c_4 - s_{23}s_4r_{31}$$

$$y = -r_{12}c_1c_{23}s_4 - r_{12}s_1c_4 - r_{22}s_1c_{23}s_4 + r_{22}c_1c_4 - s_{23}s_4r_{32}$$

3.1.5 Jacobian of the Wrist Centre

The Jacobian is used to relate the joint velocities to the end-effector's velocity. It is also used to calculate the singularities within the system. The Jacobian for the CRS F3 is:

$${}^0\mathbf{J} = \begin{bmatrix} \mathbf{j}_1 & \mathbf{j}_2 & \mathbf{j}_3 & \mathbf{j}_4 & \mathbf{j}_5 & \mathbf{j}_6 \end{bmatrix} \quad (3.12)$$

where

$$\mathbf{j}_1 = \begin{bmatrix} s_1 s_{23} d_4 - s_1 c_2 a_2 - s_1 a_1 \\ -c_1 s_{23} d_4 + c_1 c_2 a_2 + c_1 a_1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{j}_2 = \begin{bmatrix} -c_1 c_{23} d_4 - c_1 s_2 a_2 \\ -s_1 c_{23} d_4 - s_1 s_2 a_2 \\ -s_{23} d_4 + c_2 a_2 \\ s_1 \\ -c_1 \\ 0 \end{bmatrix}$$

$$\mathbf{j}_3 = \begin{bmatrix} -c_1 c_{23} d_4 \\ -s_1 c_{23} d_4 \\ -s_{23} d_4 \\ s_1 \\ -c_1 \\ 0 \end{bmatrix}$$

$$\mathbf{j}_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -c_1 s_{23} \\ -s_1 s_{23} \\ c_{23} \end{bmatrix}$$

$$\mathbf{j}_5 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ c_1 c_{23} s_4 + s_1 c_4 \\ s_1 c_{23} s_4 - c_1 c_4 \\ s_{23} s_4 \end{bmatrix}$$

$$\mathbf{j}_6 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -(c_1 c_{23} c_4 - s_1 s_4) s_5 - c_1 s_{23} c_5 \\ -(s_1 c_{23} c_4 + c_1 s_4) s_5 - s_1 s_{23} c_5 \\ -s_{23} c_4 s_5 + c_{23} c_5 \end{bmatrix}$$

3.2 Singularities of the Manipulator

As mentioned above, the Jacobian relates the joint velocities to the end-effector velocities. For certain values of the joint angles, the end-effector's velocity will be undefined. At these undefined regions, the manipulator is in what is called a singularity. A singularity represents a configuration of the manipulator where it instantaneously loses a DOF. By taking the determinant of the Jacobian and equating it to *zero*:

$$|\mathbf{J}| = 0 \quad (3.13)$$

the singularity conditions can be found. For the CRS F3, the determinant of the Jacobian is:

$$|\mathbf{J}| = s_5 c_3 (a_2 c_2 + a_1 - d_4 s_{23}) \quad (3.14)$$

Therefore, the singularities are $s_5 = 0$, $c_3 = 0$, and $a_2 c_2 + a_1 = d_4 s_{23}$.

The singularities for the CRS F3 are visually depicted in Figures 3.2 and 3.3. In Figure 3.2, an example of the singular configuration governed by $s_5 = 0$ is shown. In this configuration, the axis of joint 6 is collinear with the axis of joint 4. An example of the singular configuration governed by $c_3 = 0$ is depicted in Figure 3.3. In this configuration, link three is in line with link five. The singular configuration governed by $a_2c_2 + a_1 = d_4s_{23}$ cannot be reached due to the CRS F3's joint limits and link lengths, therefore, this singularity may be omitted.

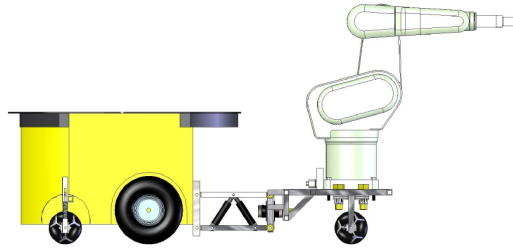


Figure 3.2: Singular Configuration: $s_5 = 0$

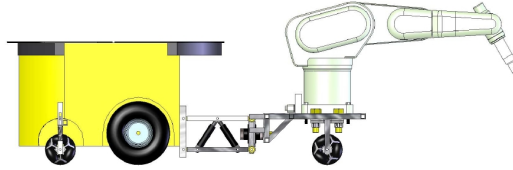


Figure 3.3: Singular Configuration: $c_3 = 0$

3.3 Mobile Base

The mobile base has 2-DOF, it is able to move forward and rotate about the vertical axis. This is achieved through its differential steering, where one wheel spins with a different velocity than the other. Due to the wheel layout, the mobile base is not able to move laterally. This is known as a nonholonomic constraint. The mobile base is not able to move any direction, unlike an omni-directional vehicle (i.e., it has a limited range of motion).

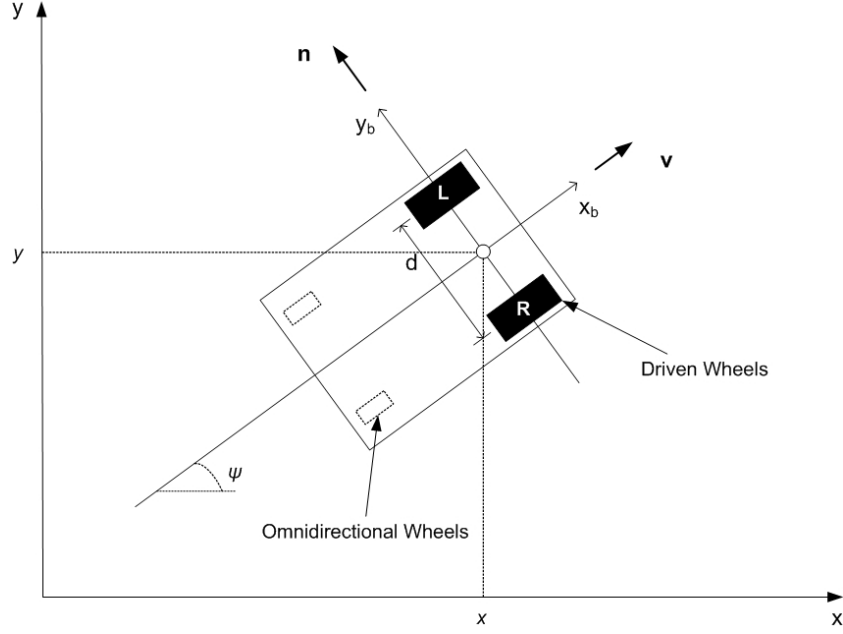


Figure 3.4: Mobile Base

A simplified model of the mobile base is shown in Figure 3.4. The base is nonholonomic and therefore governed by the state equations [2]:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{\theta}_R \\ \dot{\theta}_L \end{bmatrix} = \begin{bmatrix} \frac{r}{2} \cos \psi & \frac{r}{2} \cos \psi \\ \frac{r}{2} \sin \psi & \frac{r}{2} \sin \psi \\ \frac{r}{l} & -\frac{r}{l} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\theta}_R \\ \dot{\theta}_L \end{bmatrix} \quad (3.15)$$

where $(\dot{x}, \dot{y}, \dot{\psi})$ is the velocity of the base, r is the radius of the drive wheels, l is the length between the wheel centres, and $\dot{\theta}_R$ and $\dot{\theta}_L$ are the rotational velocities of the right and left wheels, respectively. It should be noted that this model assumes that there is no slip in the wheels.

From this model, several things become apparent. If the left and right wheel speeds are positive and equal, the mobile base will move solely in the x direction. If the right wheel speed is greater than the left wheel speed, the mobile base will turn left. If the

left wheel speed is greater than the right wheel speed, the mobile base will turn right. If the left and right wheel speeds are opposite to one another, the mobile base will simply spin about the centre point between its wheels.

Chapter 4

Control Algorithm

4.1 Design Process

Mobile-manipulator systems are difficult to control, especially as their degrees-of-freedom (DOF) increase. Many researchers have proposed algorithms for their control to some degree of success. However, most of these algorithms lack in one aspect or another. It is important to understand the current problems before developing an improved algorithm. This allows for valuable lessons to be learned. The current problems are listed below, they are not withstanding to all papers:

1. **Physical singularities are not avoided:** Much of the research has been focused on solving the inverse kinematics of the mobile-manipulator systems with no regard to singularity avoidance. Singularity avoidance is very important because there is a loss in mobility once a singularity is reached, defeating one of the main purposes of mobilizing a manipulator.
2. **Algorithmic singularities:** Since most mobile manipulators are modeled as one system, using either an extended or augmented Jacobian, algorithmic singularities occur. These singularities do not represent physical singularities. They just represent areas where the algorithm breaks down, which makes it then

inoperable in these areas.

3. **Predefined task trajectory used:** In order to test their control algorithms, most researchers use a predefined task trajectory for their end-effector. In real-life application, a predefined end-effector trajectory would most likely not be used. The end-effector trajectory would come from an operator or, in the case of autonomous systems, from sensory feedback and would change dynamically as the operator/system sees fit. This allows the system to be flexible and allow for on-the-fly changes to be made.
4. **More than one input device is used:** If operator input is used to control a mobile-manipulator system, two joysticks are usually used, one joystick for controlling the manipulator and another for controlling the mobile base. This requires great coordination, skill, and concentration. It might even require a second operator. An improved method of control would require less skill and less concentration and be of equal or greater effectiveness. This would allow the operator to do a better job and possibly let him/her complete other tasks in the process or more complex assignments.
5. **Algorithms are tested through simulation rather than on a physical prototype:** In order to prove that the algorithm is valid it should be implemented on a real system. This is the only way that it can truly be verified. Simulations always have assumptions. Things may get overlooked or implementing certain ideas may not even be possible, therefore simulation results may not be accurate. Therefore, it is important to test control algorithms on physical prototypes.

The problems above serve as a clear indication of some of the requirements for a novel control algorithm for mobile manipulators and a guide as to what to avoid. Based upon these problems and sound design principles, a series of precepts have been

developed during the early design phase of the control algorithm. These precepts create a framework on which the algorithm is built upon. They represent how the algorithm should be formulated. The precepts are listed below. They correspond numerically to the above listed problems.

1. **Manipulator should be in a state of high manipulability:** The manipulator is responsible for carrying out fine tasks with its end-effector. In certain configurations, it is much better at doing so. It would therefore be ideal if the manipulator would be in such a position most of the time. The operator would then have the most effective use of the manipulator. In the same sense, the manipulator should also avoid being in singular configurations, as the loss of DOF, makes the mobile manipulator less useful.
2. **Separation of the manipulator and mobile base:** The manipulator and mobile base should be separated in two senses. In the first sense, the manipulator and mobile base should be modeled separately. Algorithmic singularities occur when both systems are modeled as one (through the use of one Jacobian). Therefore, it is important to model the manipulator on its own and the mobile base on its own. A way of modeling the dependence of the two on each other will be required. In the second sense, the motion of the manipulator and mobile base should be separated as much as possible. Due to the effects of noise and complexity of moving both systems at the same time, the manipulator should move only when it needs to move and the mobile base should move only when it needs to move. This also leaves the operator from focusing on both elements of the system and being concerned with how both will react. It also improves accuracy. The manipulator has been designed for achieving fine motions, while the mobile base can only achieve coarse motions. The operation of the mobile base with the manipulator will cause the end-effector to lose some of its precision.

3. **Real-world application:** The control algorithm is intended to be used in real-world applications and not just as a theoretical example. Therefore, it should be designed so that it can handle real-world scenarios. A source of dynamic input should be used, rather than predefined coordinates along a path, to control the mobile manipulator. The system should be able to handle on-the-fly changes.
4. **Ease of control:** The control algorithm should make it easy for the mobile manipulator to be controlled. Rather than two input devices, one input device should be used. The operator should not have to worry about both systems, only the end-effector, as this is the primary tool. The control algorithm should be intuitive, meaning that it should have a good sense of the user's intentions. This is difficult to do since a machine cannot predict what a user intends to do. However, if the control algorithm is made simple enough, then the intuitive nature may seem apparent. Minimal amount of training should be required.
5. **Physical prototype:** In order to verify the control algorithm, it should be implemented on a physical mobile manipulator, rather than simulated. This will also help clearly and concisely show how it works.

Two additional precepts were also implemented. These were not based on the current problems of mobile manipulators, but based on good design principles. They are:

6. **Configurable algorithm:** The control algorithm should be configurable. The user should be able to specify certain options which tune the system to his/her preferences. Different scenarios/tasks will also require different options and these should all be adjustable so the system can be adaptable.
7. **Expandable algorithm:** Many types of different mobile manipulators exist from terrestrial ones to underwater ones. It is important that the control algorithm be expandable to all these mobile manipulators. It will specifically be

implemented on Jasper, but it should be able to be applied to other mobile manipulator systems as well. Therefore, the possibility of using a system with a different configuration should be taken into consideration.

4.1.1 Concept Generation

Concept generation is the process of generating ideas. It is important to generate as many ideas as possible when designing a control algorithm. Each idea has its own advantages and disadvantages. These advantages and disadvantages can be analyzed and together these ideas can be built upon. The eventual goal is to create the best control algorithm possible. Concept generation greatly improves the chances of doing so. The ideas that were generated in the design process of the novel control algorithm are presented here. They include ideas regarding the input device, manipulator control, and mobile base control/unification.

4.1.1.1 Input Device

The 6-DOF joystick by RSI Research Ltd. has been chosen as the input device for the mobile manipulator. This is due to a number of reasons.

First and foremost, the 6-DOF joystick has the same amount of DOF as the manipulator. This will make direct control of the manipulator easy. Second, 6-DOF is easy to understand since it is used by all people to position and orient objects on a daily basis. It is the highest amount of DOF that one can have without being redundant (in the 3D world).

Not many 6-DOF input devices exist. This is the only known 6-DOF joystick of its kind having been based on a three branch parallel mechanism. Other 6-DOF input devices are primarily used for gaming consoles. These, however, are wireless and do not have the accuracy of the RSI 6-DOF joystick. If, for example, a wireless joystick is released it will fall and tumble on the ground causing unpredictable motion

in the manipulator. With a joystick that is physically connected to a base, if the joystick handle is released, it will spring back to its home position and stop, causing a predictable motion in the manipulator.

The joystick has the added appeal in that it only requires use of one hand and allows for intuitive control. Since it is attached to the operator's hand and moves in 3D space like the manipulator's end-effector, the operator can see his/her hand motions as a direct reflection of the manipulator's end-effector's movements.

The technical specifications of the joystick can be found in Chapter 2.

4.1.1.2 Manipulator Control

Separation of motion is required of the control algorithm. Portions of the control will involve direct joystick control of the manipulator's end-effector with the mobile base being stationary. It is best to develop the easiest and most useful method of direct control as the manipulator will be responsible for completing a majority of the tasks without the assistance of the mobile base.

Type of Control

Two types of control are possible: position and velocity control. Each has its own advantages and disadvantages. The two types of control are summarized as follows:

1. **Position Control:** This involves the direct control of the manipulator's position. Changes in the joystick position and orientation are converted into changes in the manipulator's position and orientation.

Advantages: The advantage of such a method is that the end-effector's position change is proportional to the joystick's position change. This makes it easy for the operator to control. It is as if the operator's hand is at the end-effector, making it intuitive by nature. Furthermore, this method is very easy to implement.

Disadvantages: Since the speed of the end-effector will be constant during operation, large translations will take a much longer time than small translations. The entire timing of the system will vary and such a system will be difficult to synchronize.

2. **Velocity Control:** This involves the direct control of the manipulator's velocity. Changes in the joystick position and orientation are converted into changes in the manipulator's velocity (angular velocity in regards to orientation). The further the joystick is from a designated home position, the faster the end-effector moves.

Advantages: The advantage of this is opposite to that of position control. The speed of the end-effector changes as the operator sees fit and the speed can be changed every iteration. Each iteration can take the same amount of time. This means that this type of control is much easier to synchronize as there is no dependence on how far the end-effector has moved.

Disadvantages: In a control scheme like this, fine control may be difficult. When humans perform pick-and-place operations, they base movements on their hand's position and not their hand's velocity. This would make this type of control not intuitive in nature. In addition, velocity control is difficult to implement on the CRS F3 articulated arm.

Based on the above advantages and disadvantages, position control was chosen as the type of control for the manipulator. It is both intuitive and simple to implement. The ease of use of such a controller is worth the trade off in difficulty in synchronization.

Method of Control

Now that the type of control has been chosen, a method for controlling the manipulator with the joystick is necessary. There are many different ways to control solely

the manipulator with the joystick, presented as follows:

1. **World Coordinates:** One way of controlling the end-effector is to use the joystick's position and orientation to translate directly into the end-effector's position and orientation with respect to the manipulator's base frame. The joystick's position changes would translate intuitively into the manipulator's translations, pushing forward on the joystick would result in a forward movement of the manipulator, pushing right on the joystick would result in a right movement of the manipulator, etc. This is illustrated in Figure 4.1.

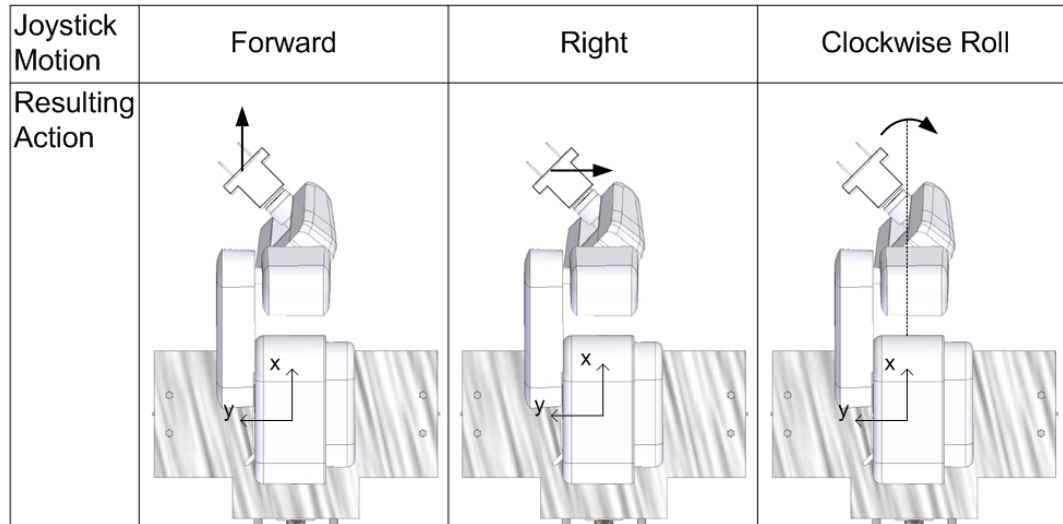


Figure 4.1: Control using World Coordinates

Advantages: The advantage of this is that determining how the position of the joystick affects the end-effector's movement is very easy to understand.

Disadvantages: It is very difficult to understand how the orientation of the joystick affects the end-effector's orientation. Since the rotation occurs about the manipulator's base frame, it may be hard to achieve certain desired motions without an expert understanding of rotational transformations. In addition, achieving the same orientation change will depend on where the end-effector is located. Therefore, each time a new joystick

motion will be necessary. Also the operator will need to keep track of where the end-effector is in regards to the world frame.

2. **Tool Coordinates:** Another way of controlling the end-effector is to use the tool frame of the manipulator. The tool frame is fixed to the end-effector. In this control method, the joystick's position and orientation would directly translate into the end-effector's position and orientation with respect to the tool frame. Pushing the joystick forward would result in the end-effector moving forward, pushing right on the joystick would result in a right movement of the manipulator, etc., all with respect to the current orientation of the end-effector. It is as if the operator would be sitting on the end-effector. This is shown in Figure 4.2.

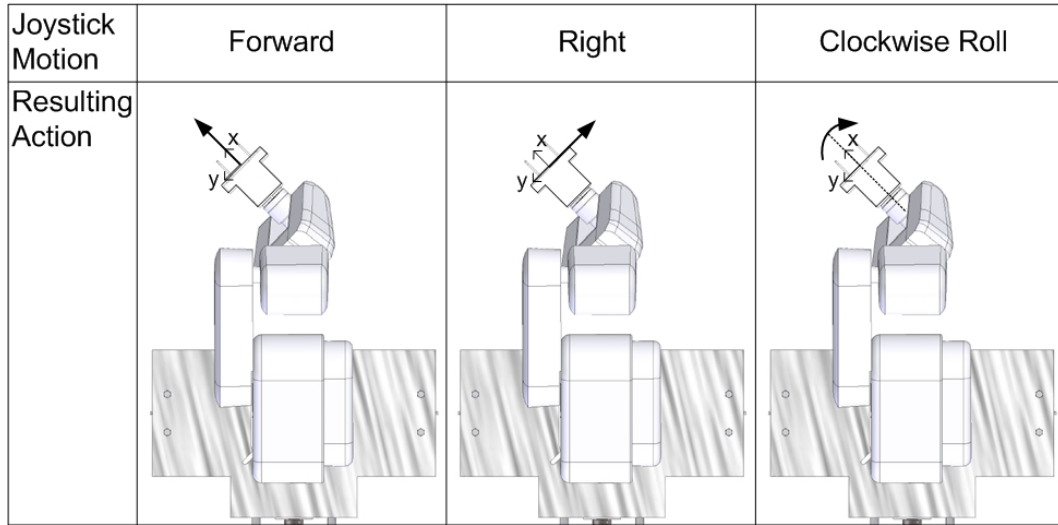


Figure 4.2: Control using Tool Coordinates

Advantages: This control strategy makes it very easy for the operator to understand how position and orientation changes of the joystick will affect the position and orientation of the end-effector. It is very intuitive. Orientation changes are always the same, regardless of the end-effector's position, unlike world coordinate control. If, for example, the operator wants to

rotate the end-effector so that it can turn a screw, all he/she has to do is roll the joystick. Changes in roll, pitch, and yaw of the joystick directly translate to changes in roll, pitch, and yaw of the end-effector.

Disadvantages: There are no major disadvantages to this method.

3. **Joint Angles:** The possibility exists of controlling each of the joints individually. In this control scheme, each of the 6-DOF of the joystick's motion would correspond to one of the joints. For example, rotation of the joystick about the vertical axis could result in rotation of joint 1 and rotation of the joystick about the horizontal axis could result in rotation of joint 2.

Advantages: The advantage of this is that precise joint control is available.

There may be situations where the operator just wants to control one of the joints and leave the rest stationary.

Disadvantages: This type of control is not intuitive at all. One would need an expert understanding of the manipulator's layout/kinematics in order to control it. This type of control, however, is not useless, it may be practical for manipulators that have different configurations (three revolute joints and three prismatic joints) or that have fewer DOF.

Tool coordinate control has been chosen for direct control of the end-effector using the joystick. It is the easiest method of control allowing both the position and orientation to be controlled intuitively. It also allows the joystick's 6-DOF to perfectly correspond to the manipulator's 6-DOF, 3-DOF used for positioning and 3-DOF used for orienting.

Method of Implementation

With the above, World Coordinate Control, Tool Coordinate Control, and Joint Angle Control, there are two possible methods of implementation: Absolute Control and

Incremental Control.

1. **Absolute Control:** Absolute control means that the joystick's position and orientation with respect to its resting or home frame, is proportional to the manipulator's position and orientation with respect to its ideal position. So when the joystick is pushed fully forward, the manipulator will be fully extended forward and remain there. Once the joystick returns to its home position, the manipulator also returns to its ideal position.

Advantages: This type of control makes it easy to understand how the joystick movements affect the manipulator as the manipulator is in essence a mirror of the joystick. It is very intuitive.

Disadvantages: Major problems exist with this control scheme. First, the joystick is small in comparison to the manipulator. The scaling factor that transforms the joystick's position to the manipulator's position would have to be very large so that when the joystick is at its limit, the manipulator would be too. This would make the manipulator very sensitive to movements and would therefore, require a very steady hand. Accidental movements of the joystick could easily cause damage.

Additionally, the joystick is designed to spring back, so the operator cannot just let the joystick go and expect the manipulator to remain in place. If this were to happen, the manipulator would move back into its home position as well. As a result, the operator must always keep his/her hand where he/she wants the manipulator's end-effector to be. This could lead to great discomfort and fatigue.

The joystick is also limited from achieving certain orientations due to its physical constraints. This would mean that not all orientations of the manipulator could be achieved through this type of control.

Lastly, the ideal position of the manipulator would have to be near the centre of its workspace, since the joystick's resting/home position is at the centre of its workspace. This would allow for all regions of the joystick's workspace to be utilized.

2. **Incremental Control:** Incremental control means that the joystick's position and orientation change with respect to its home position constantly increments the end-effector's position and orientation, during each iteration. For example, if the joystick is pushed 1 cm forward, the manipulator will move 1 cm forward during the first iteration of the control loop, 1 cm forward during the second iteration, and so on. Once the joystick is returned to its home position, the manipulator stops moving.

Advantages: This control strategy is also intuitive and easy to use. When the operator returns the joystick back to its home position, the manipulator stops moving. This is an important safety feature. If the operator should leave the joystick or let it go accidentally, the manipulator would stop moving. Additionally, the sensitivity of the manipulator to the joystick's motion can easily be adjusted with scaling factors and is not limited by the size of the joystick and its physical limits.

Disadvantages: In order to use this type of control, the user must have a good sense of the time of each control iteration as each motion is updated every iteration. However, if the operator has a problem with this, a scaling factor can be used to limit the motion during each iteration.

Based on the above advantages and disadvantages, incremental control was chosen as the method of implementation. With its inherent safety features and ease of use, it makes it ideal for controlling a mobile manipulator.

4.1.1.3 Mobile Base Control/Unification

Now that sole control of the manipulator has been established, a method of controlling the mobile base is required as well. It is important to understand that the mobile base serves the function of transporting the manipulator. It may also be used for carrying items that the manipulator needs (i.e., tools) or items that need to be stored. While the mobile base transports the manipulator, the manipulator can be arranged into its most useful configuration. So it is possible for the manipulator to move when the mobile base is moving (deemed as unified motion).

The mobile base uses velocity control; both the left and right wheel velocities are controlled. Velocity control has been used for the mobile base since the differential equations describing the mobile base's motion deal with velocity. If position is needed, the velocity can easily be numerically integrated.

Absolute control is used to control the wheel speeds. The mobile base's controller is given the desired wheel velocities every iteration.

Three methods for controlling the mobile base were reviewed, they are discussed and analyzed herein:

1. **Continuous Base Movement:** This method of control involves placing no restrictions on the mobile base. The mobile base moves in order to bring the manipulator into a configuration of as high manipulability as possible. An optimization routine is used to continuously find this configuration. As a result, this will likely result in constant mobile base movement. This type of control is not novel.

Advantages: The advantage of this method is that there is no need to monitor singularities. The manipulator is always in an optimal configuration, therefore, it will keep itself within its ideal workspace.

Disadvantages: The mobile base does not have high positional accuracy (when

compared to the manipulator). If it is constantly moving, then it will lessen the accuracy of the end-effector. The accuracy of the end-effector will also be reduced due to vibrations caused by the dynamic interaction of the moving base and moving arm.

2. **Discrete Base Movement:** In this method of control, the mobile base moves only when it needs to. When the manipulator extends outwards and approaches a singularity, the mobile base begins to move. The mobile base moves in conjunction with the manipulator in such a way as to bring the manipulator into a more suitable configuration. The end-effector's position and orientation remain constant with respect to the world frame. Once the system has achieved the more suitable configuration, the base stops moving and the manipulator moves on its own again until it begins to approach another singularity. The mobile manipulator system, therefore, *inchworms* its way to its target. This type of control is not novel either.

Advantages: The advantage of this control method is that it keeps the manipulator in a configuration of high manipulability. In addition, the translation of the end-effector in regards to the world frame is solely achieved by the manipulator's motion. This allows for great positioning accuracy. The motion of the mobile base and manipulator are therefore separated.

Disadvantages: This method of control does not allow fluid motion of the end-effector. The end-effector moves and stops repeatedly. This can cause operations to be really slow.

3. **Continuous and Discrete Base Movement:** As in the previous method, the mobile base moves only when it needs to. Once the manipulator approaches a singularity, the mobile base moves. The mobile base and manipulator move in such a way as to keep the end-effector moving in its last direction (before

having approached a singularity). Once the end-effector reaches its desired position (indicated by the operator), the manipulator and mobile base move in unison bringing the manipulator into an ideal configuration, while maintaining the position of the end-effector constant with respect to the world frame. This type of control is novel.

Advantages: This control method keeps the manipulator in a useful configuration. It also provides fluid motion of the end-effector. The translation of the end-effector in regards to the world frame is solely achieved by the manipulator's motion, when the manipulator is required to perform a task. This makes it very accurate.

Disadvantages: The only disadvantage of this control method is the need to monitor singularity conditions.

The Continuous and Discrete Base Movement control method was selected as the method for controlling the mobile base. It combines the features of the other two proposed methods in that it places the manipulator into a configuration of high manipulability when that configuration is desired. It also separates the manipulator from the mobile base when the manipulator is to be used for fine operations. Lastly, it moves the end-effector in a fluid motion.

4.2 Overview

An overview of the basics of the control algorithm is presented herein. The control algorithm has several inputs: the pose of the 6-DOF joystick and the current joint angles of the manipulator. It can be described by three states. These states govern the behaviour of the mobile-manipulator system. The states, described below, and illustrated in Figure 4.3, occur sequentially.

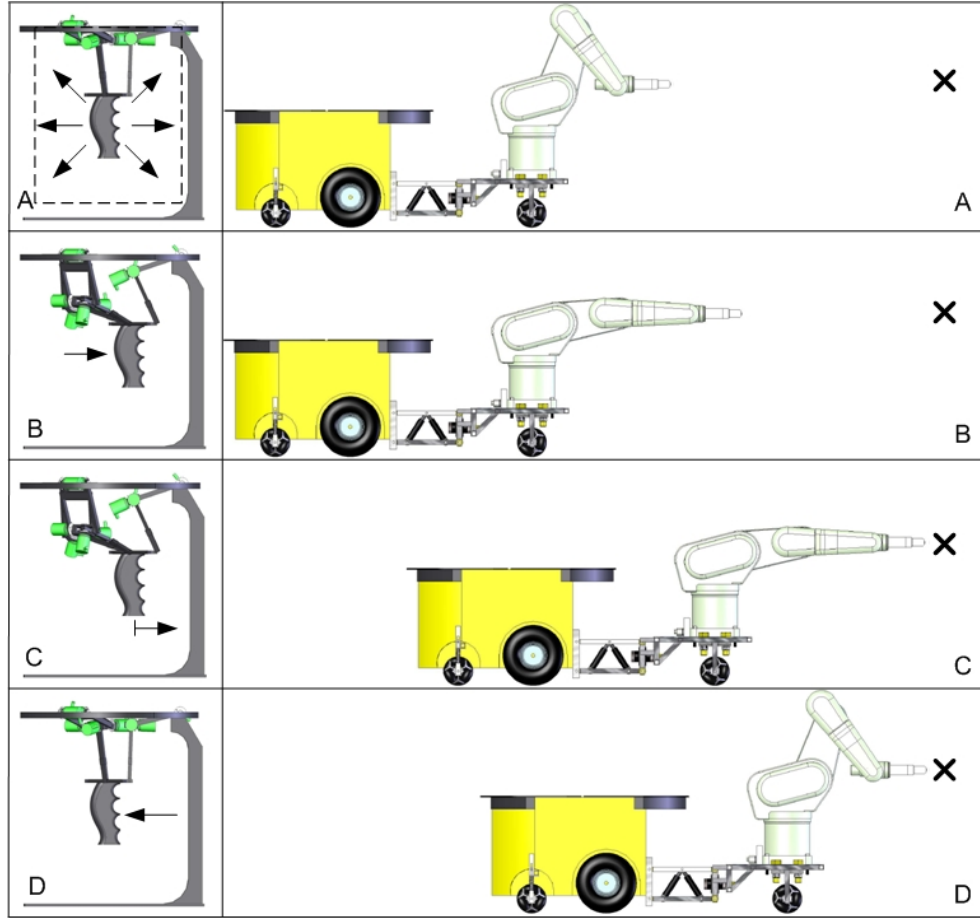


Figure 4.3: Control States

The control algorithm uses knowledge of the manipulator's singularities to determine the movement of the arm and base. In State 1, the manipulator is far from singularities as shown in Section A of Figure 4.3. Here, the joystick controls the manipulator alone. Joystick motion is converted directly into motion of the manipulator. Once the manipulator approaches a singularity (as shown in Section B), the mobile manipulator enters State 2. In this state, the mobile base moves the manipulator forward in the direction that the manipulator was traveling before the singularity was reached (as shown in Section C). Once the user pulls the joystick back to its home position State 3 is entered. The manipulator and mobile base then move in a unified fashion so as to keep the end-effector in its current position (with respect to the tool frame) and bring the manipulator into a more preferred configuration (one of higher manipulability) as

shown in Section D.

The details of the each State are presented in the following sections.

4.3 State 1: Manipulation



Figure 4.4: State 1

The mobile manipulator is in its first state when the manipulator's end-effector is within its work envelope, far from singularities as shown in Figure 4.4. In this state, the joystick is used to control the manipulator, with the mobile base remaining completely stationary. The joystick can travel in any direction to control the manipulator. The joystick's motion is transferred to that of the tool frame. Forward and backward motion of the joystick results in forward and backward motion of the end-effector. Up and down motion of the joystick results in up and down motion of the end-effector. Left and right motion of the joystick results in left and right motion of the end-effector. The same is true in regards to rotations. All motions and all rotations are with respect to the end-effector's tool frame. In essence, the operator is placed at the tip of the end-effector. No matter how the end-effector is oriented, if the user wants to go in the direction in front of him/her, he/she just pushes the joystick forward. This type of control is very similar to that of flying a helicopter. It is illustrated in Figure 4.5. This provides for simple control, in accordance to Precept 1. The operator does not

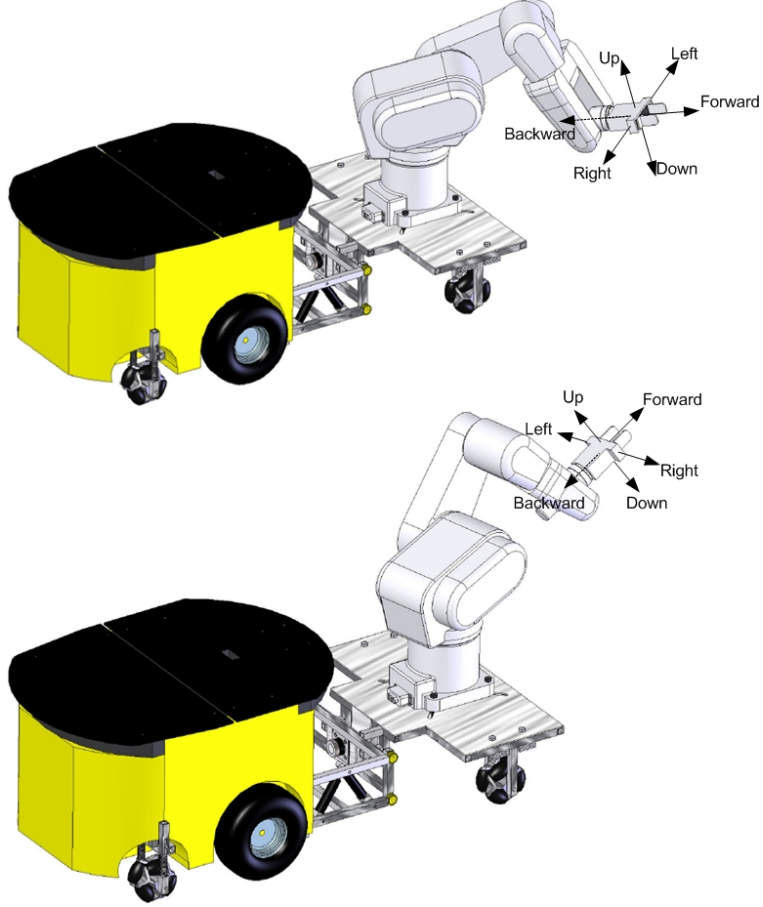


Figure 4.5: Control Example for State 1

have to think about how his/her previous actions affect the manipulator or where the end-effector is in terms of the world frame in order to control the manipulator. He/she just has to imagine him/herself on the end-effector.

The joystick's world frame, $\{J\}$, is located atop the joystick (see Figure 4.6). Another frame, called the home frame, $\{H\}$, is designated 10 cm below. The home frame designates the joystick's resting position. Both frame $\{J\}$ and $\{H\}$ are fixed in space. A frame called the current frame, $\{C\}$, is attached to the joystick's handle and moves as the handle moves. The joystick's coordinates are given with respect to the joystick's world frame. For purposes of practicality, the joystick's coordinates with respect to the home frame are better suited. Hence, the position and orientation of frame $\{C\}$ with respect to frame $\{H\}$, described using the homogeneous transform ${}^H_C\mathbf{T}$, is required.

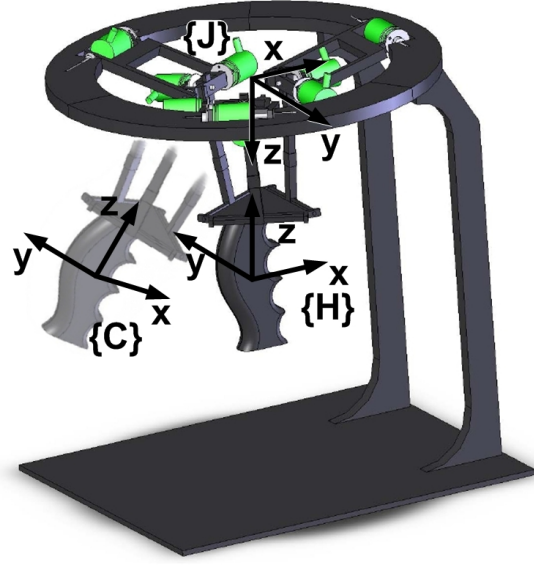


Figure 4.6: Joystick Frames

It can be found as follows:

$${}^H_C \mathbf{T} = {}^H_J \mathbf{T} {}^J_C \mathbf{T} \quad (4.1)$$

where ${}^H_J \mathbf{T}$ is the transformation matrix that relates $\{J\}$ to $\{H\}$ and ${}^J_C \mathbf{T}$ is the homogeneous transform describing the joystick's current frame with respect to the joystick's world frame.

${}^H_C \mathbf{T}$ can be written as the following transformation matrix:

$${}^H_C \mathbf{T} = \begin{bmatrix} {}^H_C \mathbf{R} & {}^H \mathbf{p}_C \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.2)$$

where ${}^H_C \mathbf{R}$ is a 3x3 rotation matrix and ${}^H \mathbf{p}_C$ is a position vector starting at the origin of frame $\{H\}$ and ending at the origin of frame $\{C\}$.

The manipulator's world frame of reference, $\{M\}$, is located at the centre of its base and its tool frame is located at the middle tip of the end-effector (where the fingers are connected). The frames are highlighted in Figure 4.7.

The joystick's position, ${}^H \mathbf{p}_C$, relative to the fixed home frame, is used in a position

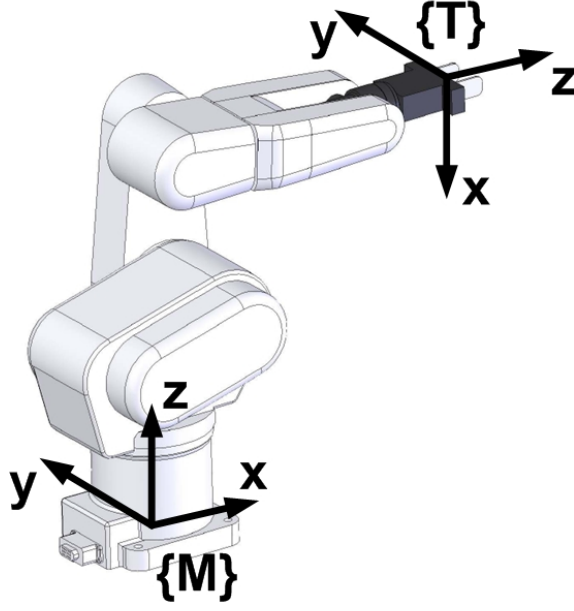


Figure 4.7: Manipulator Frames

controller that controls the translational position of the manipulator's end-effector. The further the joystick is from the home frame, the further the manipulator translates. This is accomplished by mapping the end-effector position, ${}^M\mathbf{P}_T$, using ${}^H\mathbf{P}_C$. Figure 4.8 shows how these two frames are related.

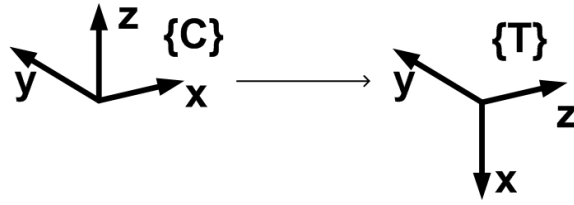


Figure 4.8: Mapping of Joystick Frame to Tool Frame

As one can see the frames are not in-line. Each axis should be pointing in the same direction so that the forward direction of the joystick will mean that the end-effector moves in the forward direction, rather than downwards. To align the frames, or in other terms to properly map ${}^H\mathbf{P}_C$ to ${}^M\mathbf{P}_T$, a simple 90° rotation about the y -axis is required:

$${}^M\mathbf{P}_T = \mathbf{R}_y(90^\circ) {}^H\mathbf{P}_C \quad (4.3)$$

Rather than just using these coordinates to direct the manipulator, an opportunity for customization appears. A matrix, containing scaling factors, can be used to convert the coordinates:

$$\begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} \alpha_1 & 0 & 0 \\ 0 & \alpha_2 & 0 \\ 0 & 0 & \alpha_3 \end{bmatrix} \begin{bmatrix} {}^H p_x \\ {}^H p_y \\ {}^H p_z \end{bmatrix} \quad (4.4)$$

where the scaling factors α_1 , α_2 , and α_3 can be adjusted based on user preferences. For example, if the scaling factors are set below 1, then the joystick loses sensitivity. However, if the scaling factors are set above 1, then the joystick's sensitivity increases and a slight motion of the joystick will produce a large motion of the end-effector. Also, the scaling factors do not have to be identical. It might be beneficial to have reduced sensitivity in the vertical direction for example, if one is operating in a mine with a low ceiling. If the operator raises the joystick too fast, the end-effector has the potential of hitting the ceiling. Ideally, these scaling factors should be updateable in real-time.

The inverse displacement solution of the arm is then used to calculate the necessary joint angles. However, this is handled by the manipulator's controller.

The orientation of the joystick, ${}^H_C \mathbf{R}$, with respect to the home frame is used to control the orientation of the end-effector directly. This is done by setting the rotation matrix of the manipulator at frame $\{6\}$, with respect to the base frame $\{0\}$, to ${}^H_C \mathbf{R}$:

$${}^0_6 \mathbf{R} = {}^H_C \mathbf{R} \quad (4.5)$$

This results in the end-effector having an identical orientation to that of the joystick. This becomes beneficial as it, in effect, places the operator's hand at the manipulator's end-effector.

Additionally, joint limits are used to ensure that the end-effector does not collide with the ground, the mobile base, and the connection platform, despite the operator's

commands.

The 6-DOF joystick is a passive device. With no human support, the joystick handle hangs (with springs holding it in place) and it does not necessarily return to the exact same position each time. As it is spatial, it is also very difficult for a user to bring the joystick into its home position exactly. In this set-up, even the smallest joystick deviation from $\{H\}$ would affect the manipulator's movement. This is problematic. When the operator brings the joystick to the home position, he/she expects the end-effector to remain stationary. In order to achieve this, and reduce the fine motor skills required by the operator in order to bring the joystick to exactly $\{H\}$, it is best to implement a deadband filter.

The deadband filter allows the operator to bring the joystick into a small region around the home position. This way the operator can stop moving the end-effector and either think about what else needs to be done or do something else. The position deadband is shown in Figure 4.9. The boxed region represents the deadband. If the centre of the triangular end-effector platform is within this region, no translational motion of the end-effector is registered. The size of region is customizable.

This applies to orientation as well. Small accidental shifts in orientation can cause the end-effector to change its orientation as well. Again, in order to reduce the demand on the fine motor skills of the operator (and make his/her life much easier), a deadband filter should be instituted on orientation. Only deliberate changes of the joystick's orientation should change the orientation of the end-effector and when the joystick is oriented straight up there should be no end-effector motion. The orientation-deadband filter differs from the position-deadband filter as small deviations in orientation should not affect the end-effector even when the joystick is far from $\{H\}$. An example of the orientation deadband filter is shown in Figure 4.10. The joystick's orientation must be greater than the limits of the regions shown for orientation to be registered by the end-effector. Again, the limits of the orientation deadband can be adjusted by the

operator to suit their preference.

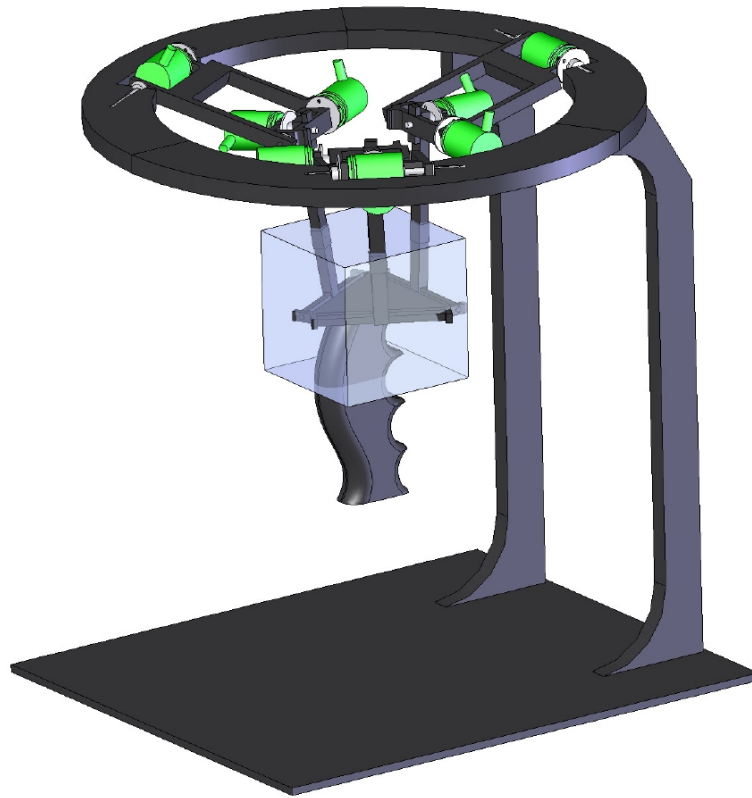


Figure 4.9: Position Deadband

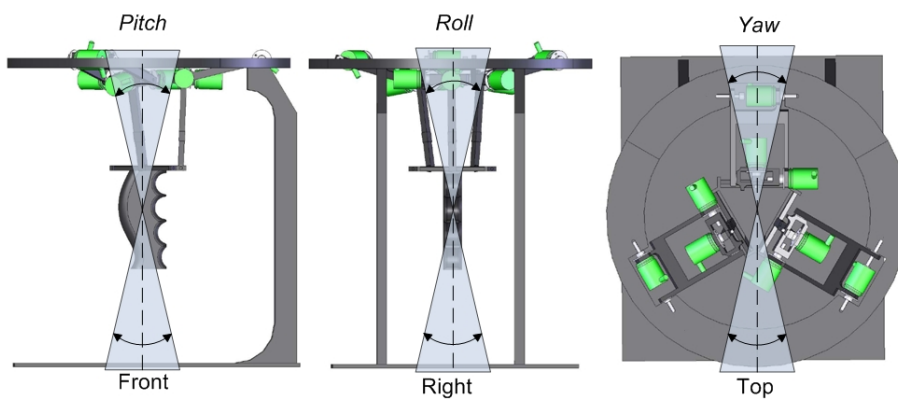


Figure 4.10: Orientation Deadband

4.4 State 2: Approaching a Singularity

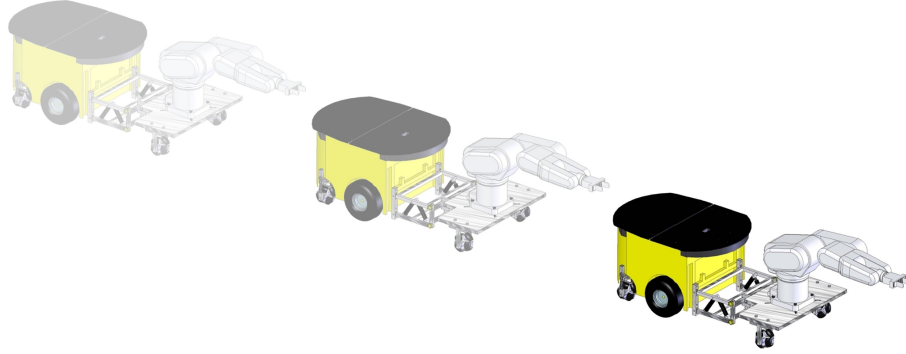


Figure 4.11: State 2

As opposed to State 1, State 2 occurs when the manipulator's end-effector approaches a singularity, a configuration where the manipulator instantaneously loses a DOF. The singularities of the manipulator have been found by setting $|\mathbf{J}| = 0$ as shown in Chapter 3. They are governed by the following three equations:

$$s_5 = 0 \tag{4.6}$$

$$c_3 = 0 \tag{4.7}$$

$$a_2c_2 + a_1 = d_4s_{23} \tag{4.8}$$

where a_1 , a_2 , and d_4 are known link lengths and offsets, and c_{ij} and s_{ij} , denote $\cos(\theta_i + \theta_j)$ and $\sin(\theta_i + \theta_j)$, respectively.

As mentioned before, the condition outlined in equation (4.8) is not achievable for the CRS F3. Also, since the purpose of the algorithm is to position the wrist centre, the condition outlined in equation (4.6) is not a concern since it deals with the wrist orientation. Therefore, only the condition of equation (4.7) is monitored. When the angle of θ_3 approaches the singularity condition described in equation (4.7) within a specified value of β (left to be defined by the user), the system moves so as to keep the end-effector moving in its last direction with respect to the horizontal plane. To

achieve this, the mobile base moves in conjunction with the manipulator. This is done through the use of optimization. It should be noted that with respect to State 1, the joystick is still in the same position or further from its home position.

State 2 is shown in Figure 4.3 in both Sections B and C. In Section B, the joystick is moved completely forward and the manipulator has approached a singular configuration as the operator desires the end-effector to move towards the target, designated by an $\mathbf{x}$. Section C shows what happens once the singularity is reached, the mobile base travels forward as the end-effector's last motion before approaching the singularity was forward. In order to keep the mobile manipulator within this state, the operator can either leave the joystick in its current position or push it further forward. Pushing the joystick further forward will cause the velocity of the mobile base to increase.

To determine the movement of the manipulator and mobile base during State 2 an optimization needs to be performed, since there are more than 6-DOF that need to be controlled. The problem is solved in three stages: pre-optimization, optimization, and post-optimization. They are described below.

4.4.1 Pre-Optimization

The pre-optimization stage is responsible for calculating the new desired wrist centre position with respect to the world frame. This is visually outlined in Figure 4.12.

By knowing the components of the vector that represents the manipulator's last direction of movement (before reaching a singularity), $direct_x$ and $direct_y$, the angle, α , between this direction vector and the x -axis of the mobile base frame can be found using the following equation:

$$\alpha_{1,i} = \text{atan2}(direct_y, direct_x) \quad (4.9)$$

Now that α is known, it is possible to calculate the desired offset in the x and y

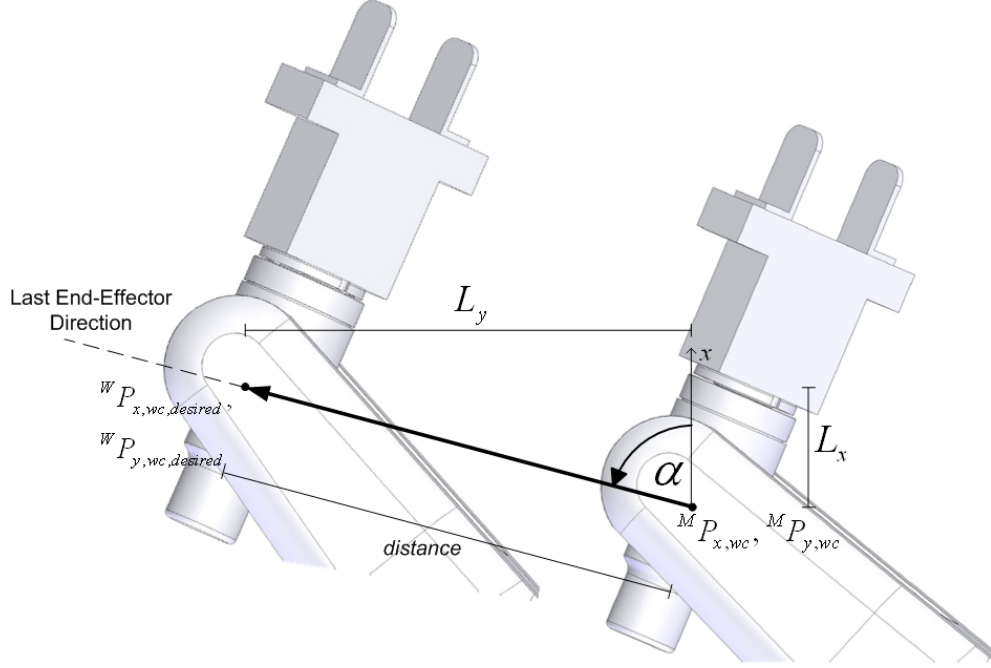


Figure 4.12: Wrist Centre Position Determination

direction. This desired offset is based on the set *distance* that the end-effector is to travel over one iteration of the optimization. This is achieved through the equations:

$$L_x = \cos(\alpha - {}^W\psi_{MB}) \times \text{distance} \quad (4.10)$$

$$L_y = \sin(\alpha - {}^W\psi_{MB}) \times \text{distance} \quad (4.11)$$

where L_x and L_y are the desired offsets of the wrist-centre in the world frame and ${}^W\psi_{MB}$ is the angle of rotation between the mobile base's frame and the world frame. This takes into account the continuing rotation of the mobile base as it moves and adjusts the angle so that the manipulator's end-effector moves in a straight line with respect to the world frame. It should be noted that frames $\{W\}$ and $\{B\}$ are coincident at the start of this algorithm.

Knowing these desired offsets, the new wrist centre position (with respect to the world frame) can be found by adding the desired offsets to the current wrist centre position

$({}^M P_{x,wc}, {}^M P_{y,wc})$ using:

$${}^W P_{x,wc,desired} = {}^M P_{x,wc} + L_{offset} + L_x \quad (4.12)$$

$${}^W P_{y,wc,desired} = {}^M P_{y,wc} + L_y \quad (4.13)$$

where L_{offset} is the distance between the mobile base frame and the manipulator's frame.

The *reach* is also found as:

$$reach = \sqrt{{}^M P_{x,wc}^2 + {}^M P_{y,wc}^2} \quad (4.14)$$

It represents the horizontal distance between the wrist centre and the manipulator's base frame. It, along with the desired wrist centre position (with respect to the world frame), will be used in the optimization routine described below.

4.4.2 Optimization

Optimization is the technique of finding the minimum (or maximum) value of a function. This is done by systematically varying the values of the variables of a function until the desired function value is achieved. The function that is being optimized is denoted as the objective function. The variables can also be constrained by both linear and nonlinear functions. These constraints limit the solution space and govern how the optimization problem is solved.

Various optimization routines exist and some are better than others at solving different types of problems. The problem at hand is a highly nonlinear one due to the many trigonometric functions involved. As a result, an equality constrained sequential quadratic programming (SQP) method called DONLP2 [46] is used.

The optimization routine is to be run on a computer which controls the mobile-manipulator system and will play a crucial role in determining how the mobile ma-

nipulator moves. As such, requirements need to be defined such that the performance of the system is not affected or affected to a minimum. Therefore, any optimization routine used to solve the problem requires the following: 1. Precision: The routine should be as precise as possible, so that dexterity is maximized to its fullest extent. 2. Fast: The routine should also be as close to real-time as possible so that there is no lag in the system. 3. Reliable: The routine should always be able to solve the optimization problem, since if it does not, the mobile manipulator will not know how to move.

4.4.2.1 Objective Function

The three search variables are θ_{1f} , the final angle of joint 1, and $\dot{\theta}_{r,wheel}$ and $\dot{\theta}_{l,wheel}$, the velocities of the mobile base's wheels.

The objective function is as follows:

$$f(\theta_{1f}, \dot{\theta}_{lw}, \dot{\theta}_{rw}) = w_1(\theta_{1f} - \theta_{1i})^2 + w_2\sqrt{{}^W\Delta x^2 + {}^W\Delta y^2} \quad (4.15)$$

where w_1 and w_2 are weight factors, θ_{1i} is the initial angle of joint 1, and ${}^W\Delta x$ and ${}^W\Delta y$ are the x and y displacements, respectively, of the mobile base with respect to the world frame.

The “objective” in this case is to achieve the desired motion while minimizing the movement of the arm ($\theta_{1f} - \theta_{1i}$) and minimizing the motion of the base. The weight factors are variables that allow the operator to control how much arm movement and base movement to minimize with respect to each other. During testing they were both set to 1.

4.4.2.2 Constraints

The constraints govern how the optimization problem is solved. They create boundaries in the solution space.

There are many constraints to this problem. The most important one being the non-holonomic constraint of the mobile base described by equation (3.15). The equations give the velocity components of the mobile base and the rate of its rotation, $\dot{x}$, $\dot{y}$, and $\dot{\psi}$, from the right and left wheel speeds, $\dot{\theta}_{r,wheel}$ and $\dot{\theta}_{l,wheel}$. In order to control the mobile-base, the right and left wheel speeds are sent to the mobile-base's controller. Therefore, the inputs to these equations are fine for control purposes. However, the position and orientation of the mobile base as an output would be better suited. Implementing equation (3.15) requires numerically calculating the derivatives of these differential equations. By definition, the derivative of a function is equivalent to the change in the function over the change in time, as the change in time approaches zero:

$$\frac{df(t_0)}{dt} = \lim_{t \rightarrow 0} \frac{f(t) - f(t_0)}{t - t_0} = \lim_{t \rightarrow t_0} \frac{\Delta f}{\Delta t} \quad (4.16)$$

As a result, equation (3.15) may be written as:

$$\lim_{\Delta t \rightarrow 0} \frac{\Delta^W x_B}{\Delta t} = \lim_{\Delta t \rightarrow 0} \frac{r}{2} \cos^W \psi_B \left(\frac{\Delta \theta_{r,wheel}}{\Delta t} + \frac{\Delta \theta_{l,wheel}}{\Delta t} \right) \quad (4.17)$$

$$\lim_{\Delta t \rightarrow 0} \frac{\Delta^W y_B}{\Delta t} = \lim_{\Delta t \rightarrow 0} \frac{r}{2} \sin^W \psi_B \left(\frac{\Delta \theta_{r,wheel}}{\Delta t} + \frac{\Delta \theta_{l,wheel}}{\Delta t} \right) \quad (4.18)$$

$$\lim_{\Delta t \rightarrow 0} \frac{\Delta^W \psi_B}{\Delta t} = \lim_{\Delta t \rightarrow 0} \frac{r}{l} \left(\frac{\Delta \theta_{r,wheel}}{\Delta t} - \frac{\Delta \theta_{l,wheel}}{\Delta t} \right) \quad (4.19)$$

However, in order to implement this model of the mobile-base, the equations must be made discrete. This involves approximating the derivative with very small values of Δt :

$$\frac{df(t_0)}{dt} \approx \frac{\Delta f}{\Delta t} \quad (4.20)$$

This results in the following approximation of the mobile base model:

$$\frac{\Delta^W x_B}{\Delta t} \approx \frac{r}{2} \cos^W \psi_B \left(\frac{\Delta \theta_{r,wheel}}{\Delta t} + \frac{\Delta \theta_{l,wheel}}{\Delta t} \right) \quad (4.21)$$

$$\frac{\Delta^W y_B}{\Delta t} \approx \frac{r}{2} \sin^W \psi_B \left(\frac{\Delta \theta_{r,wheel}}{\Delta t} + \frac{\Delta \theta_{l,wheel}}{\Delta t} \right) \quad (4.22)$$

$$\frac{\Delta^W \psi_B}{\Delta t} \approx \frac{r}{l} \left(\frac{\Delta \theta_{r,wheel}}{\Delta t} - \frac{\Delta \theta_{l,wheel}}{\Delta t} \right) \quad (4.23)$$

where Δt is a very small value. This introduces some error into the model, however, if the time interval is small enough, the error will be insignificant.

However, to simplify the format of this model, Δt may be removed as it is a common factor in all equations. This results in the following set of equations, that give the mobile base's position and orientation rather than velocity and rate of change of orientation.

$$\Delta^W x_B \approx \frac{r}{2} \cos^W \psi_B (\Delta \theta_{r,wheel} + \Delta \theta_{l,wheel}) \quad (4.24)$$

$$\Delta^W y_B \approx \frac{r}{2} \sin^W \psi_B (\Delta \theta_{r,wheel} + \Delta \theta_{l,wheel}) \quad (4.25)$$

$$\Delta^W \psi_B \approx \frac{r}{l} (\Delta \theta_{r,wheel} - \Delta \theta_{l,wheel}) \quad (4.26)$$

Since the model's input are the right and left wheel speeds, $\dot{\theta}_{r,wheel}$ and $\dot{\theta}_{l,wheel}$, $\Delta \theta_{r,wheel}$ and $\Delta \theta_{l,wheel}$ can be found through the approximation of equation (4.20):

$$\Delta \theta_{r,wheel} = \dot{\theta}_{r,wheel} \Delta t \quad (4.27)$$

$$\Delta \theta_{l,wheel} = \dot{\theta}_{l,wheel} \Delta t \quad (4.28)$$

again where Δt is very small.

This model finds the displacements of the mobile base over the time span of Δt , to find the absolute position, the displacements must be added to the actual current

position (${}^Wx_{B0}$, ${}^Wy_{B0}$, and ${}^W\psi_{B0}$):

$${}^Wx_B = {}^Wx_{B0} + \Delta^Wx_B \quad (4.29)$$

$${}^Wy_B = {}^Wy_{B0} + \Delta^Wy_B \quad (4.30)$$

$${}^W\psi_B = {}^W\psi_{B0} + \Delta^W\psi_B \quad (4.31)$$

where ${}^Wx_{B0} = 0$ and ${}^Wy_{B0} = 0$.

Being able to find the mobile base's position based on its wheel velocities, allows for the new wrist centre position, (${}^WP_{x,wc,var}$, ${}^WP_{y,wc,var}$), to be found. Figure 4.13 illustrates how this new wrist centre position is found.

The optimization varies the wheel velocities of the mobile base. This results in the mobile base having various positions (${}^WX_B$, ${}^WY_B$) and orientations (${}^W\psi_B$). Using these results and the distance between the midpoint of the wheels and the manipulator's base, the position of the manipulator's base with respect to the world frame (${}^WX_{M1}$, ${}^WY_{M1}$) can be found:

$${}^WX_{M1} = {}^WX_B + L_{offset} \cos({}^W\psi_B) \quad (4.32)$$

$${}^WY_{M1} = {}^WY_B + L_{offset} \sin({}^W\psi_B) \quad (4.33)$$

Using the *reach* that was calculated in equation (4.14), the new wrist centre position is found with the optimization routine varying $\theta_{1,f}$:

$${}^WP_{x,wc,var} = {}^WX_{M1} + reach \times \cos({}^W\psi_B + \theta_{1,f}) \quad (4.34)$$

$${}^WP_{y,wc,var} = {}^WY_{M1} + reach \times \sin({}^W\psi_B + \theta_{1,f}) \quad (4.35)$$

The variable *reach* ensures that the new wrist centre position is the same distance away from the manipulator's base frame as it currently is. This keeps the manipulator outstretched and in effect keeps joints 2 and 3 fixed. The forward displacement

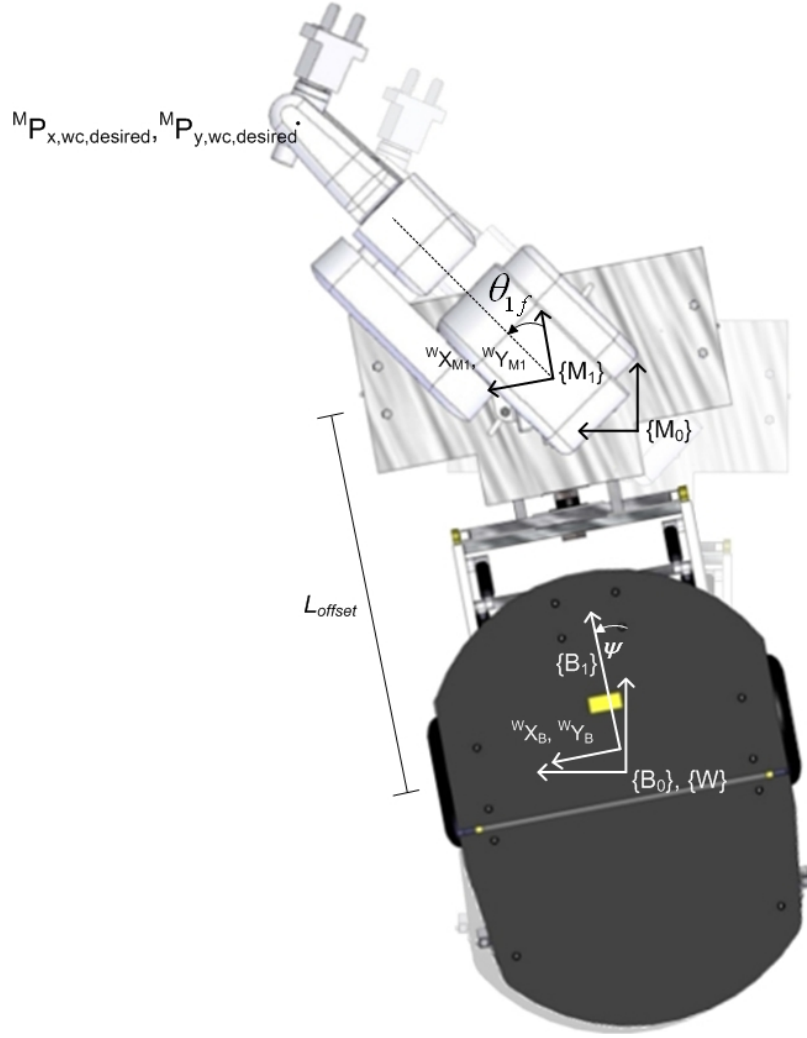


Figure 4.13: Determination of New Wrist Centre Position

solution of the wrist centre could have been used, however this would have been more costly computationally (i.e., it would have required more memory resources and would have taken a longer time to solve).

The optimization routine keeps varying $\dot{\theta}_{r,wheel}$, $\dot{\theta}_{l,wheel}$, and $\theta_{1,f}$. Consequently, the new wrist centre position may not be equivalent to the desired wrist centre position calculated using equation (4.13) prior to the optimization. In order for the optimiza-

tion to have any value, the following constraints must be enforced:

$${}^wP_{x,wc,var} - {}^wP_{x,wc,desired} = 0 \quad (4.36)$$

$${}^wP_{y,wc,var} - {}^wP_{y,wc,desired} = 0 \quad (4.37)$$

In addition to the nonholonomic constraints, constraints on the manipulator's joint angles and the mobile base's wheel velocities need to be enforced.

Table 4.1 shows the range of each of the six joints as well as their respective maximum velocities.

Table 4.1: CRS F3 Joint Constraints [41]

Axis	Range	Maximum Velocity
Joint 1	$\pm 180^\circ$	$240^\circ/\text{s}$
Joint 2	-135° to $+45^\circ$	$240^\circ/\text{s}$
Joint 3	$\pm 135^\circ$	$240^\circ/\text{s}$
Joint 4	$\pm 180^\circ$	$375^\circ/\text{s}$
Joint 5	$\pm 135^\circ$	$300^\circ/\text{s}$
Joint 6	± 4096 turns	$375^\circ/\text{s}$

Constraints have been imposed on the joint angles making sure that they do not exceed those listed in Table 4.1. Each iteration of the optimization has its own possible joint limits that span a smaller range. This is because the joints have a maximum velocity and each iteration only spans a time of Δt . As a result, the maximum and minimum possible joint movement can be found as follows:

$$\text{Max. Possible Joint Movement} = \text{Current Joint Angle} + \text{Max. Joint Vel.} \times \Delta t \quad (4.38)$$

$$\text{Min. Possible Joint Movement} = \text{Current Joint Angle} - \text{Max. Joint Vel.} \times \Delta t \quad (4.39)$$

This is shown in Figure 4.14. The overall joint limits always take precedence because mechanically, the joints cannot move any further. This is illustrated in Figure 4.15.

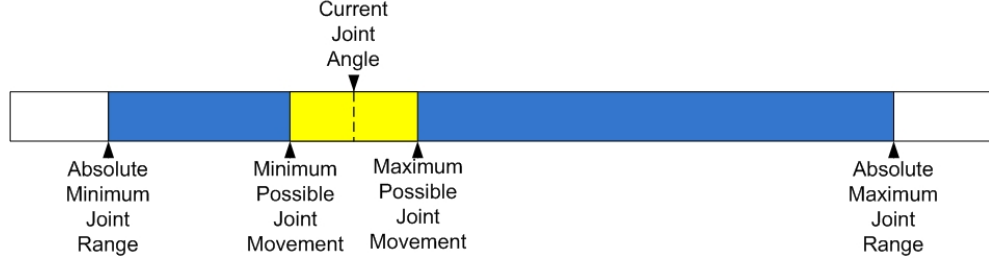


Figure 4.14: Maximum Possible Joint Movement



Figure 4.15: Precedence of Maximum Joint Range

It should be noted that acceleration has not been incorporated into the joint limit constraints in order to keep the optimization simple and much faster. This can be added in the future to improve the accuracy of the constraints.

Unlike the manipulator's joints, the mobile base's wheels do not have a range of rotation, they only have a maximum velocity. The maximum velocity of each wheel is ± 2.48 rev/s. Therefore, the following constraints are applied to the optimization as well:

$$-2.48 \frac{rev}{s} \leq \dot{\theta}_{r,wheel} \leq 2.48 \frac{rev}{s} \quad (4.40)$$

$$-2.48 \frac{rev}{s} \leq \dot{\theta}_{l,wheel} \leq 2.48 \frac{rev}{s} \quad (4.41)$$

Again, the wheels' accelerations are not incorporated.

4.4.3 Post-Optimization

The optimization outputs the most ideal $\dot{\theta}_{r,wheel}$, $\dot{\theta}_{l,wheel}$, and θ_1 , that satisfy the listed constraints and minimize the objective function. The angles θ_2 and θ_3 are also known. They remain constant throughout State 2 in order for the manipulator to remain outstretched. These three joint angles along with the wheel speeds, place the wrist centre in its new position. If θ_4 , θ_5 , and θ_6 remain constant, then the end-effector will not have kept its current orientation with respect to the world frame, as demonstrated in Figure 4.16.

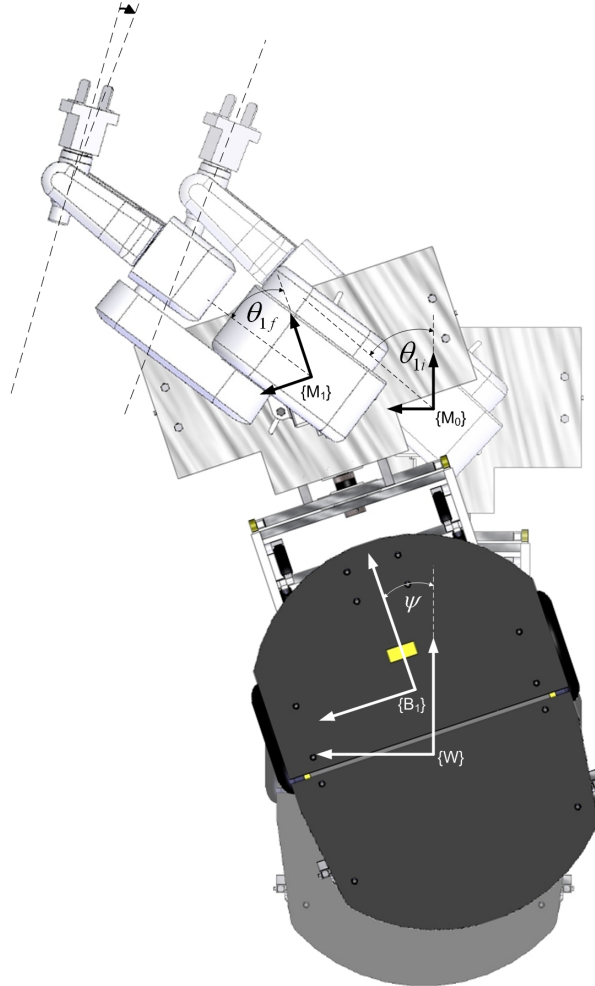


Figure 4.16: Error in End-Effector Orientation

To keep the end-effector's current orientation with respect to the world frame, the end-

effector's orientation needs to be rotated about the z -axis by $({}^W\psi_B + (\theta_{1f} - \theta_{1i}))$ in the clockwise direction. This adjusts for the end-effector's orientation change which is due to the rotation of the mobile-base and θ_1 . In order to achieve this, the initial orientation of the end-effector, ${}^0\mathbf{R}_{Current}$ is rotated through:

$${}^0\mathbf{R}_{Desired} = \mathbf{R}_{z,rot} {}^0\mathbf{R}_{Current} \quad (4.42)$$

where ${}^0\mathbf{R}_{Desired}$ is the desired end-effector orientation and where $\mathbf{R}_{z,rot}$ is defined as follows:

$$\mathbf{R}_{z,rot} = \begin{bmatrix} \cos(-({}^W\psi_B + (\theta_{1f} - \theta_{1i}))) & -\sin(-({}^W\psi_B + (\theta_{1f} - \theta_{1i}))) & 0 \\ \sin(-({}^W\psi_B + (\theta_{1f} - \theta_{1i}))) & \cos(-({}^W\psi_B + (\theta_{1f} - \theta_{1i}))) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.43)$$

The best way to solve for θ_4 , θ_5 , and θ_6 is to find ${}^3\mathbf{R}$ and find its inverse displacement solution. This is achieved by:

$${}^3\mathbf{R}_{Desired} = {}^3\mathbf{R}_6 {}^0\mathbf{R}_{Desired} \quad (4.44)$$

${}^3\mathbf{R}$ can be found by taking the transpose of ${}^0\mathbf{R}$ as shown in:

$${}^0\mathbf{R}^T = {}^3\mathbf{R} \quad (4.45)$$

where ${}^0\mathbf{R}$ is:

$${}^0\mathbf{R} = \begin{bmatrix} c_1 c_{23} & -c_1 s_{23} & s_1 \\ s_1 c_{23} & -s_1 s_{23} & -c_1 \\ s_{23} & c_{23} & 0 \end{bmatrix} \quad (4.46)$$

and its transpose is:

$${}^3_0\mathbf{R} = \begin{bmatrix} c_1 c_{23} & s_1 c_{23} & s_{23} \\ -c_1 s_{23} & -s_1 s_{23} & c_{23} \\ s_1 & -c_1 & 0 \end{bmatrix} \quad (4.47)$$

Knowing that:

$${}^3_6\mathbf{R}_{Desired} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad (4.48)$$

θ_4 , θ_5 , and θ_6 can be found through:

$$\theta_5 = \text{atan2} \left(\pm \sqrt{1 - r_{23}^2}, r_{23} \right) \quad (4.49)$$

$$\theta_4 = \text{atan2} \left(\frac{r_{33}}{\sin(\theta_5)}, \frac{-r_{13}}{\sin(\theta_5)} \right) \quad (4.50)$$

$$\theta_6 = \text{atan2} \left(\frac{-r_{22}}{\sin(\theta_5)}, \frac{r_{21}}{\sin(\theta_5)} \right) \quad (4.51)$$

θ_5 must be found first because both θ_4 and θ_6 rely on it for their solutions.

The major problem with the use of optimization is that it usually takes a long time. If the optimization takes too long, a bottleneck may occur in the controller. This can be a significant problem, especially if the optimization time is greater than Δt . This would signify that the manipulator would wait for the optimization in order to proceed with movement. In the current case, the optimization parameters needed to be adjusted to increase the speed of the optimization.

4.5 State 3: Ideal Configuration

State 2 ends and State 3 begins when the joystick is moved back to its home position, as shown in Section D of Figure 4.3. As this happens, the manipulator is placed into an ideal configuration, preferably an isotropic configuration [47]. An isotropic

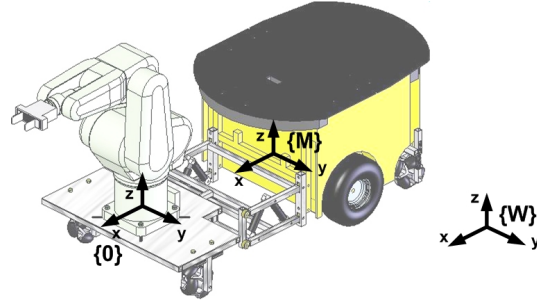


Figure 4.17: Frames of Mobile Manipulator

configuration is one of high dexterity, where the condition number of the manipulator is found to be unity. The condition number, k , can be found through:

$$k = \|\mathbf{J}\| \|\mathbf{J}^{-1}\| \quad (4.52)$$

Both the manipulator and mobile base move with respect to one another while keeping the end-effector's position and orientation constant with respect to the world coordinate frame, $\{W\}$, shown in Figure 4.17. All this is done by relating the frame of the mobile base, $\{M\}$, to the world coordinate frame, and then by relating $\{M\}$ to frame $\{0\}$ through a simple transform.

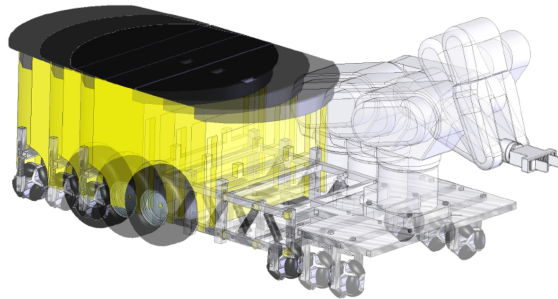


Figure 4.18: State 3

This behavior is clearly demonstrated in Figure 4.18 (and Section D of Figure 4.3). The manipulator's end-effector has remained stationary, but the configuration of the manipulator has changed. The behavior converts the manipulator from an unuseful singular configuration to a useful one, which now allows the operator to easily work on the target.

It should also be noted that if the operator commands the end-effector to move straight up vertically and the manipulator enters a singular configuration, once the operator returns the joystick to its home position, the mobile base will not be able to compensate for this, as the desired motion is purely out of the horizontal plane.

Since this is a highly non-linear problem and redundancy is involved, optimization will also be used. The DONLP2 [46] is used again.

As before, the solution can be separated into three parts: pre-optimization, optimization, and post-optimization.

4.5.1 Pre-Optimization

The first step is to define an ideal pose of the manipulator. This is the desired pose that the manipulator should be brought into during State 3. If the mobile base cannot bring the manipulator into this pose, then it should bring the manipulator as close to this pose as possible. This is done through the use of an optimization routine.

The ideal pose can be an isotropic pose (a pose of high manipulability) or it can be any other user defined pose. The option has been left to the operator, in accordance to Precept 6, giving more flexibility to the operator allowing him/her to control what the ideal pose is in order to suit the situation both before and during operation. The ideal pose used during testing and implementation is illustrated in Figure 4.19. It represents a pose of high manipulability.

Unlike State 2, the new wrist centre position does not need to be calculated. The wrist centre should remain where it is. Therefore, the position of the wrist centre

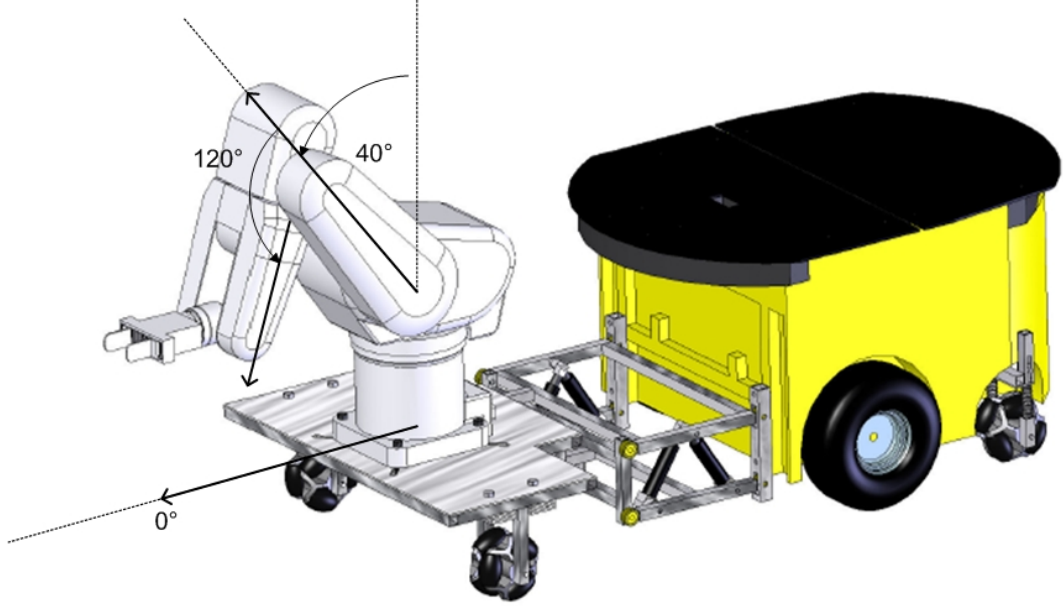


Figure 4.19: Ideal Pose

$(P_{x,wc}, P_{y,wc}, P_{z,wc})$, with respect to the world frame, remains constant.

4.5.2 Optimization

The optimization routine is responsible for bringing the manipulator closer to the ideal configuration. To do this, the optimization routine will systematically vary the left and right wheel speeds of the mobile base ($\dot{\theta}_{rw}, \dot{\theta}_{lw}$) and the first three joint angles of the manipulator ($\theta_{1,f}, \theta_{2,f}, \theta_{3,f}$) in order to minimize the objective function. While being varied, these five variables will also be constrained.

4.5.2.1 Objective Function

The objective function used is:

$$f(\theta_{1,f}, \theta_{2,f}, \theta_{3,f}, \dot{\theta}_{rw}, \dot{\theta}_{lw}) = \begin{aligned} & (\theta_{1,ideal} - \theta_{1,f})^2 + (\theta_{2,ideal} - \theta_{2,f})^2 + (\theta_{3,ideal} - \theta_{3,f})^2 \end{aligned} \quad (4.53)$$

where $\theta_{1,ideal}$, $\theta_{2,ideal}$, $\theta_{3,ideal}$ are the joint angles of the ideal pose, as defined above. It is quite evident that minimizing this objective function will only occur if the manipulator's joints $(\theta_{1,f}, \theta_{2,f}, \theta_{3,f})$ come close to the ideal joint angles. Hence, the manipulator will be brought close to the ideal pose.

4.5.2.2 Constraints

While the objective function is minimized, a few constraints must be put into place. Again the constraints keep the variables within realistic boundaries.

The most important constraint keeps the wrist centre position constant with respect to the world frame. To satisfy this constraint, the left and right wheel speeds are varied to find the position of the mobile base, $({}^Wx_B, {}^Wy_B, {}^W\psi_B)$ in accordance to equations (4.24) to (4.31). This again is the nonholonomic constraint. It is included as part of the current constraint. The mobile base's position and orientation can be summarized by the following transform:

$${}^W_B\mathbf{T} = \begin{bmatrix} \cos({}^W\psi_B) & -\sin({}^W\psi_B) & 0 & {}^Wx_B \\ \sin({}^W\psi_B) & \cos({}^W\psi_B) & 0 & {}^Wy_B \\ 0 & 0 & 1 & {}^Wz_B \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.54)$$

where the inner rotation matrix represents a rotation of ${}^W\psi_B$ about the vertical z axis.

Once the position of the mobile base is known with respect to the world frame, the joint angles $(\theta_{1,f}, \theta_{2,f}, \theta_{3,f})$ are varied to find the wrist centre position with respect to the manipulator frame (${}^M\mathbf{P}'_{wc} = [{}^Wx'_{wc}, {}^Wy'_{wc}, {}^Wz'_{wc}]^T$). This is done through the use of the forward displacement solution of the wrist centre as described by equation (3.3).

In order to constrain the wrist centre, its position should be found with respect to

the world frame. This is done through:

$$\begin{bmatrix} {}^W\mathbf{P}'_{wc} \\ 1 \end{bmatrix} = {}^W_B\mathbf{T}_M^B\mathbf{T} \begin{bmatrix} {}^M\mathbf{P}'_{wc} \\ 1 \end{bmatrix} \quad (4.55)$$

where ${}^B_M\mathbf{T}$ represents the transformation matrix of the manipulator frame with respect to the mobile base's frame and is:

$${}^B_M\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & L_{BM} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.56)$$

where L_{BM} is the distance from the mobile base frame to the manipulator frame which in this case is 750 *mm*.

The following constraint must therefore be valid:

$$\begin{aligned} {}^Wx_{wc} &= {}^Wx'_{wc} \\ {}^Wy_{wc} &= {}^Wy'_{wc} \\ {}^Wz_{wc} &= {}^Wz'_{wc} \end{aligned} \quad (4.57)$$

This constraint makes sure that the wrist centre does not move with respect to the world frame.

In addition to the nonholonomic constraint and the constraint that maintains the wrist centre at a constant position, the system also constrains the joint angles of the manipulator and wheel velocities of the mobile base in exactly the same way as in State 2. The joint angles are constrained as described by equations (4.38) and (4.39). These constraints are based on the current joint angle position and the maximum joint velocity. The joint angles can only move so much during each iteration of the optimization (depending on the joint velocity) and they cannot exceed their maximum

specified values as illustrated in Figures 4.14 and 4.15. The joint ranges and maximum velocities are summarized in Table 4.1. The wheel velocities of the mobile base are constrained using equations (4.40) and (4.41).

In regards to the optimization, it should be noted that only the variable combinations that satisfy these constraints are valid.

4.5.3 Post-Optimization

Once the optimization is completed, it is known what values of $\dot{\theta}_{rw}$, $\dot{\theta}_{lw}$, $\theta_{1,f}$, $\theta_{2,f}$, and $\theta_{3,f}$ will keep the wrist centre at a constant position with respect to the world frame and also bring the manipulator closer to its ideal pose. If only these are changed and θ_4 , θ_5 , and θ_6 are held constant, then the end-effector will not remain in constant position and constant orientation as shown in Figure 4.20.

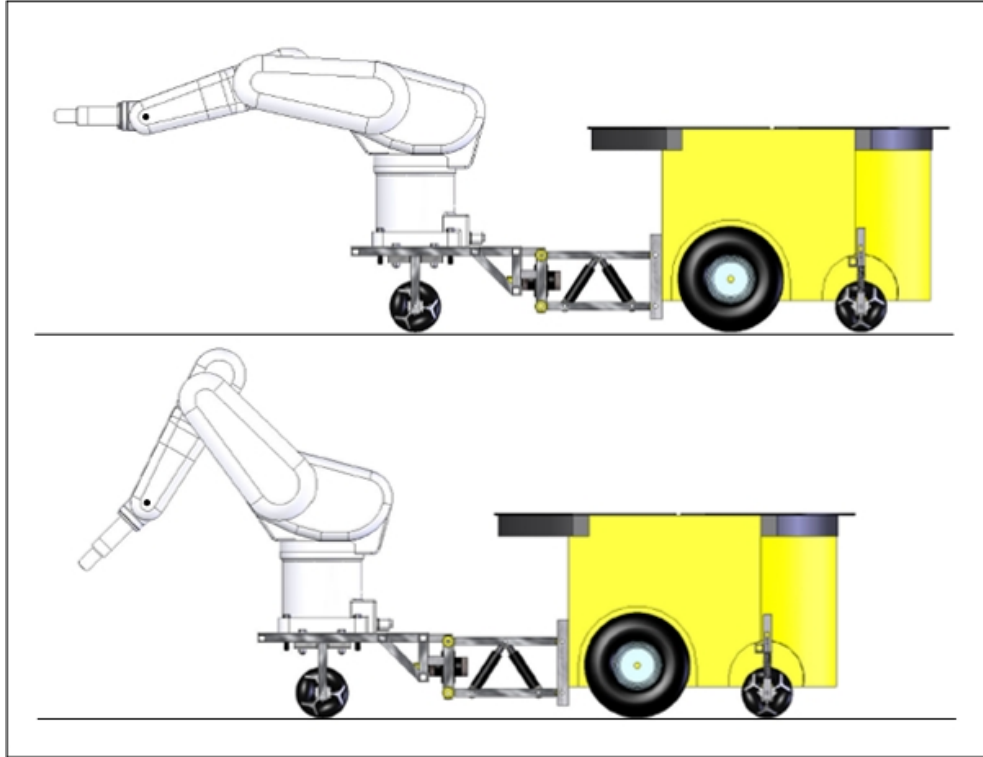


Figure 4.20: State 3 without Adjusting θ_4 , θ_5 , θ_6

The current end-effector orientation is known and can be denoted as ${}^0_6\mathbf{R}$. The initial

end-effector orientation (the orientation at the beginning of State 3) can be represented as ${}^W_6\mathbf{R}$ since frame $\{0\}$ and frame $\{W\}$ are aligned at the start. The end-effector's orientation is supposed to remain in this orientation (with respect to the world frame) throughout State 3. To find θ_4 , θ_5 and θ_6 after each iteration of the State 3 optimization, ${}^3_6\mathbf{R}$ needs to be found. It can be found through the following equation:

$${}^3_6\mathbf{R} = {}^3_0\mathbf{R} {}^0_B\mathbf{R} {}^B_W\mathbf{R} {}^W_6\mathbf{R} \quad (4.58)$$

where ${}^3_0\mathbf{R}$ is the rotation matrix describing frame $\{0\}$ with respect to the spherical wrist frame, $\{3\}$, ${}^0_B\mathbf{R}$ is the rotation matrix describing the base frame with respect to frame $\{0\}$ which is equivalent to $\{M\}$, and ${}^B_W\mathbf{R}$ is the rotation matrix describing the world frame with respect to the mobile base's frame.

The rotation, ${}^3_0\mathbf{R}$, is defined by equation (4.47) and ${}^0_B\mathbf{R}$ is:

$${}^0_B\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.59)$$

and ${}^B_W\mathbf{R}$ is:

$${}^B_W\mathbf{R} = \begin{bmatrix} \cos(-{}^W\psi_B) & \sin(-{}^W\psi_B) & 0 \\ -\sin(-{}^W\psi_B) & \cos(-{}^W\psi_B) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.60)$$

where ${}^W\psi_B$ is the rotation of the mobile base with respect to the world frame.

Once ${}^3_6\mathbf{R}$ is known, equations (4.49) to (4.51) are used to find θ_4 , θ_5 , and θ_6 .

4.6 Summary

A unified and intuitive control algorithm for the control of mobile manipulators has been developed through the use of concept generation. Several methods of control

were devised. The advantages and disadvantages of each were analyzed and the best was chosen and developed.

The developed control algorithm has been designed to allow an operator to control a 8-DOF mobile manipulator with the use of a 6-DOF joystick. The control algorithm can be described by three states. In State 1, the manipulator is far from singularities. The joystick's translation and orientation from its home position constantly increment the end-effector's position and orientation, respectively. The joystick's coordinates directly translate to those of the tool frame, in effect, placing the operator at the end-effector. Once the manipulator encounters a singularity, the mobile manipulator enters State 2. During State 2, the mobile base and manipulator move in such a way as to keep the end-effector moving in the last direction which it was moving before it approached the singularity. Once, the operator pulls the joystick back to its home position, the mobile manipulator enters State 3. State 3 dictates that the manipulator and mobile base move in unison, in a way that keeps the end-effector stationary with respect to the world frame, but brings the manipulator close to an ideal configuration. Once near this ideal configuration, the system enters State 1 again. The motions during State 2 and State 3 require optimization to be used.

Chapter 5

Implementation

The control algorithm described in Chapter 4 has been implemented on the Jasper mobile-manipulator system with success (as discussed in Chapter 6). Implementation is one of the major goals of this thesis as it helps prove/verify that the control algorithm is valid.

5.1 System Architecture

The architecture of the implemented system is shown in Figure 5.1. The nine encoders of the 6-DOF joystick are connected to the RT-Lab computer which runs QNX, a real-time operating system. A program titled Joystick Program is responsible for two tasks: calibrating the joystick and calculating the corresponding joystick position and orientation using the forward displacement solution. The joystick position and orientation are then sent to a program called the Controller program. In this program, the motion of the mobile-manipulator is resolved based on the joystick position and orientation as well as based on the manipulator's configuration (i.e., where the control algorithm of Chapter 4 is coded). If the mobile-manipulator system is in either State 2 or State 3, the Controller calls upon either the State 2 Optimization Program or the State 3 Optimization Program, respectively, to provide control. The Controller

program then sends the position of the manipulator and mobile base velocity to the Distributor program. This program is responsible for distributing data and synchronizing the system. It sends the manipulator's position and orientation (via serial cable) to the Manipulator Program located on the manipulator's controller and the mobile-base's velocities to the Mobile Base Program (via wireless Ethernet) on the mobile-base's computer. More detail in regards to these programs is provided in the following sections of this chapter.

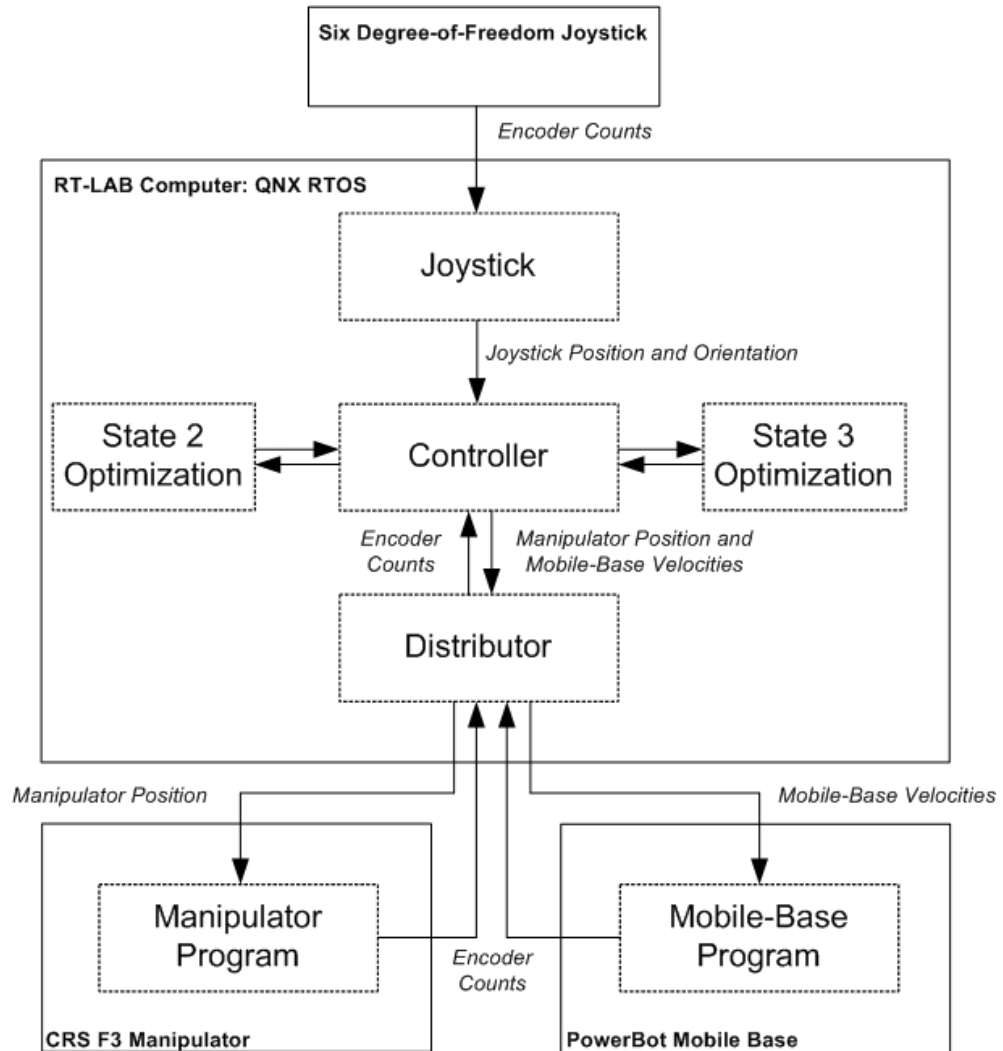


Figure 5.1: System Architecture

Functional decomposition was used in the design of the system architecture. Functional decomposition means that each part of the design has its own function. Each

of the seven programs has its own set of functions and is required to do only a certain job. This makes the system easy to implement and simple to understand. It also promotes modularity, which keeps parts of the system separated in case they need to be replaced. This is useful, if for example, the joystick were changed. This would mean that the only changes necessary to keep the system working would be to modify the Joystick Program.

5.2 Initialization

There are a total of seven programs running on the Jasper mobile-manipulator system. Five are located on the Operations Computer and the other two are located on the C500C controller and PowerBot, respectively. Due to the modular structure of the system, each program has its own function(s). To control the entire system, communication between the programs is needed. The communication of the programs proved to be one of the biggest challenges in the implementation. Three methods of communication were used: interprogram communication (IPC), communication via serial port, and wireless Ethernet communication. Before the programs begin communicating, it is important that they be initialized. The sequence of initialization here is very important and is shown in Figure 5.2.

The Mobile Base Program and Manipulator Program should be started before all other programs, however, it is acceptable if these programs start at the same time as the Joystick Program. The Mobile Base Program starts a socket server on the mobile base. Sockets are a method of communication between two programs across Ethernet. Once the server is started, the program waits for a connection from a client. The Manipulator Program opens the serial port for communication. This way it will be able to communicate with the Distributor Program.

The Joystick Program is responsible for starting the four other programs: the State 3

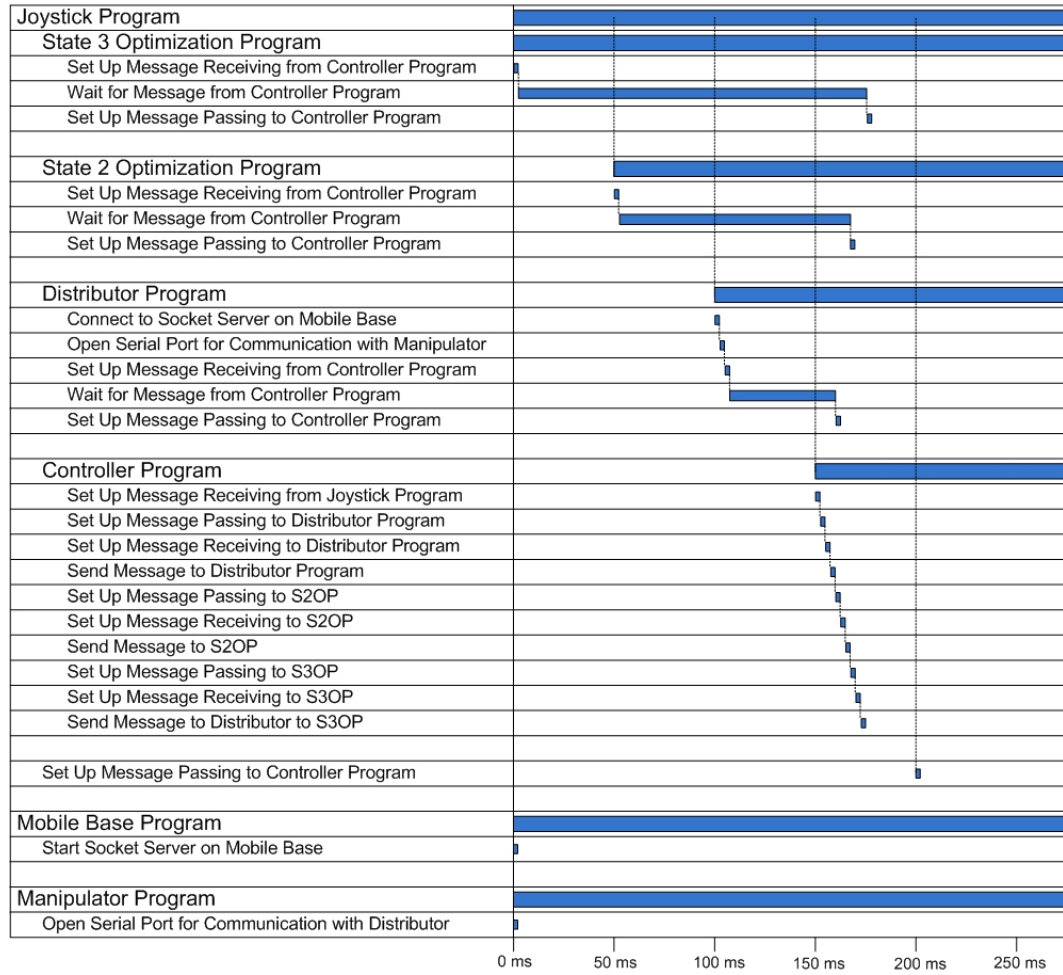


Figure 5.2: Initialization Sequence

Optimization Program, the State 2 Optimization Program, the Distributor Program, and the Controller Program. Each of these programs is started 50 *ms* apart in the respective order. This is done because there is a set order that programs need to initialize communication in and the timing facilitates this. The State 3 Optimization Program, the State 2 Optimization Program, and the Distributor Program communicate with the Controller Program through the use of QNX's message passing functions. Messages can either be received or sent. If a program is to both receive and send data, it needs to be initialized for both. The initialization of these three programs is identical with the exception of the Distributor Program which also initially connects to the socket server on the PowerBot via wireless Ethernet and then opens up the serial

port for communication with the manipulator. All these three programs receive and send information to the Controller Program. Each sets up message receiving from the Controller Program. This creates a channel for the program to receive messages on. Setting up message receiving needs to occur before setting up message passing in the Controller Program. If no channel is created (through setting up message receiving), then setting up message passing, which attaches a program to this channel, will result in an error. As a result, after these programs are initiated for message receiving, they do not immediately initiate for message passing. They wait for a message from the Controller program. This message indicates and ensures that the Controller program has set up message receiving itself with the respective programs. For this reason the Controller program is started last. It gives time for the State 3 Optimization Program, the State 2 Optimization Program, and the Distributor Program to initiate message receiving. At the end of initialization, 200 *ms* after the State 3 Optimization Program is started, message passing is set up by the Joystick Program with the Controller Program.

Fifty milliseconds was chosen as the time segment between program starts in order to allow enough time to set up the message receiving protocols. If by chance anyone of these connections fails or one connection occurs before the other, the programs will terminate or simply hang. This is a built-in safety feature. It ensures that everything is in proper working order before joystick data is transmitted to the mobile manipulator.

5.3 Joystick Program

The Joystick Program was created by Paul Sobejko at the University of Victoria [45]. It essentially does two things: it calibrates the joystick and then outputs the joystick's position and orientation.

The joystick is based on a three branch parallel mechanism. It has a total of 6-DOF. A total of nine digital optical encoders are located on the joystick. They measure the movement of the links within the branches. The encoders are incremental rather than absolute. This means that the encoders are reset to *zero* every time the Joystick Program is initiated. Since the joystick is not always physically in the exact same configuration when the Joystick Program is initiated, calibration is required to determine the encoder offsets.

Joystick calibration involves moving the joystick around randomly within the joystick's workspace for a set period of time. During these movements the encoder counts are stored in a buffer. Using the inverse displacement solution of the joystick and Levenberg-Marquardt optimization, the encoder counts in this buffer are used to calculate the encoder offsets.

Originally the Joystick Program was able to service both an analog and digital version of the joystick. The functions dealing with the analog version have been commented/removed in order to keep the size of the program small in RAM, which improves the overall program's speed. As well, the Joystick Program also had functions that allowed for fault-tolerant operation and visualization of the joystick data. Fault-tolerant operation means that the joystick would work even if an encoder failed. This is possible since the joystick has 6-DOF and nine encoders are being used. This signifies a "sensing redundancy of the third degree" [45]. As a result, the joystick could handle up to two encoder failures. These extras were also commented/removed for the same reasons.

Having been developed in 1999, the Joystick Program, written in the C programming language, was implemented on a 400 MHz Pentium-II PC running the QNX 4.24 operating system and was compiled using Watcom C 10.5 compiler. Three BB ElectronicsTM21QEC4 quadrature encoder counter cards were used to read in encoder counts. Since this system is outdated, a new computer, the Operations Computer,

was used. The Operations Computer has a 2,210 MHz AMD AthlonTM64 processor running the QNX Neutrino (v6.3.2) operating system. It also has two Sensoray 626 input/output cards that are used to read encoder counts.

In order for the Joystick Program to run on this new system, many changes had to be made. First of all, many of the standard header files and libraries that the original Joystick Program used were either outdated or no longer in existence. Therefore, each function that was no longer defined had to be ported into the new operating system. Second, since new encoder counter cards were used, the data acquisition drivers had to be updated.

The Joystick program is at the very beginning of the implementation. Its function is illustrated in Figure 5.3. The operator begins by requesting joystick calibration. Again, during the calibration, the Joystick program reads the encoder counts from the joystick (through the Sensoray 626 input/output cards). The operator then moves the joystick around in a random fashion around its workspace for a specifiable amount of iterations. The collected encoder counts are then placed into an optimization routine that finds the correct encoder offsets. At this point the program calculates the lengths of the three edges of the joystick's triangular base. In actuality, the lengths are 65.0 *mm*. However, due to imprecisions in the optimization, edge vectors between 64.0 *mm* and 66.0 *mm* can be tolerated and will give good results. If for some reason even one of the edge vectors is out of range, the program is terminated. This can be caused by calibration where the joystick movement is not random enough. The program is terminated because error in the edge vectors indicates that there will also be error in the joystick's position and orientation calculations.

Assuming a successful calibration, the Joystick Program then initializes the State 3 Optimization Program, the State 2 Optimization Program, the Distributor Program, and the Controller Program. An infinite loop is now entered. The program checks to see if the *Space bar* key has been pushed on the keyboard. In this case, the *Space*

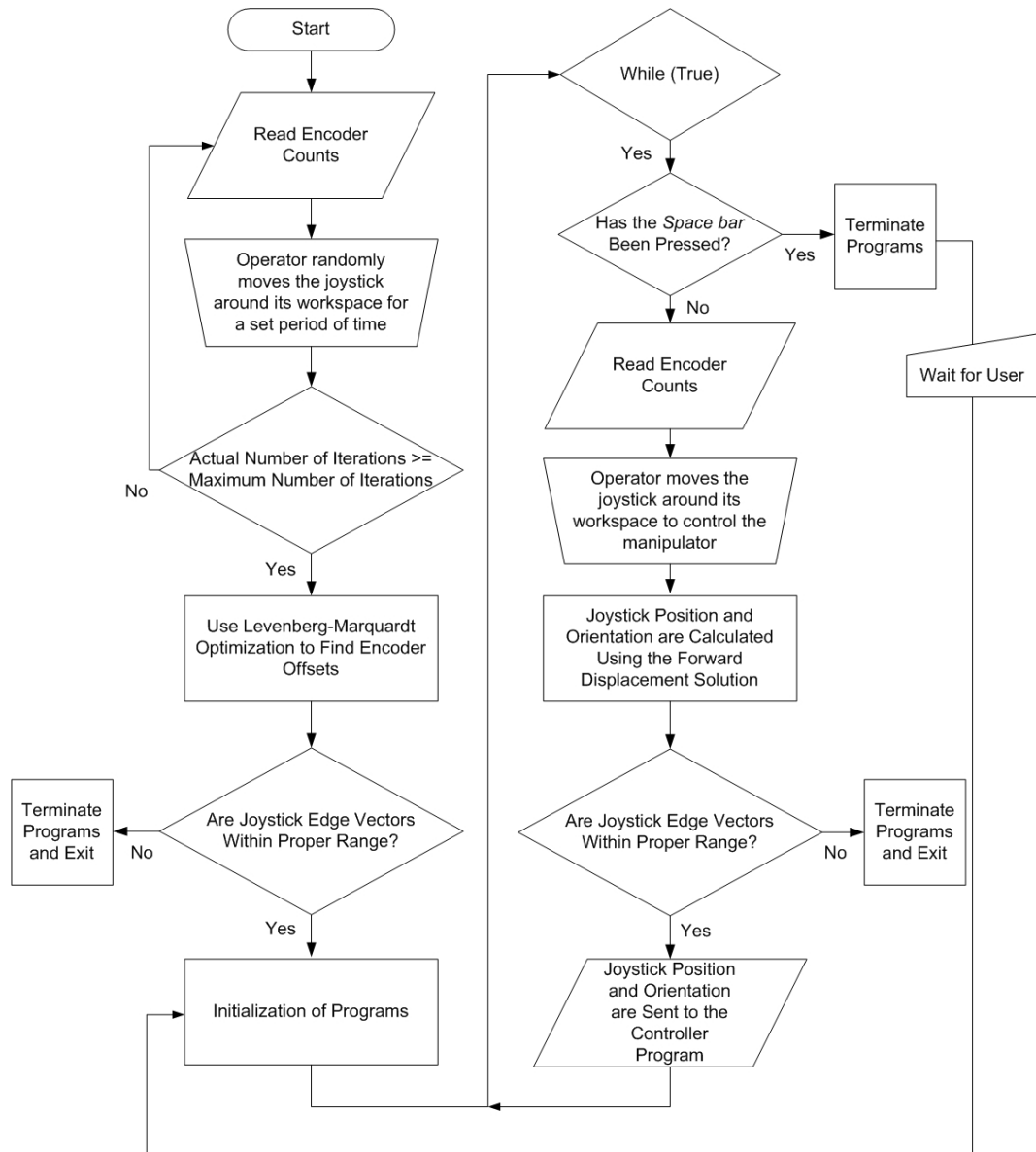


Figure 5.3: Joystick Program Overview

bar key represents an “Emergency Stop Button.” If at anytime the operator wishes to stop the system, he/she just needs to press the *Space bar* key and the Joystick Program will terminate the four programs that it initiated during its next iteration of the while loop. The Joystick Program will then wait for the operator to signal it to initialize the programs again and again it will enter the infinite loop. If the *Space bar* key is not pressed, the program reads the encoder counts while the operator moves the joystick (to control the mobile-manipulator system). The joystick position and orientation are calculated using the joystick’s forward displacement solution. Again, the program calculates whether or not any one of the edge vectors of the end-effector platform is out of range. If it is, the programs are terminated. The edge vectors may be out of range here because of poor optimization results, however, it could also be a result of screws becoming loosened on the joystick due to the constant motion or encoders slipping/not being counted fast enough. Regardless, it verifies that the Joystick Program is working correctly. Lastly, the program sends the joystick’s position and orientation (as a rotation matrix) to the Controller Program using QNX’s messaging protocol.

For more details regarding the Joystick Program please see [45].

5.4 Controller Program

The Controller Program is the *brain* of the entire system. It is where the novel control algorithm is implemented and it determines the motion of the mobile manipulator system. It has many functions: it receives joystick, manipulator, and mobile base data, it determines the last end-effector direction, it filters the joystick data, it decides based on the input and feedback how the mobile manipulator should act, and it sends coordinates to the Distributor Program. The Controller Program is written in the C programming language.

The function of the Controller Program is illustrated in Figures 5.4 and 5.5.

The Controller Program begins by setting *State* to 1. *State* is a flag variable that indicates the state the control algorithm is in. It is used to determine how the program functions. The Manipulator Program, discussed below, always brings the manipulator into a configuration of high manipulability at the beginning of control, far from singularities.

The program then sends a message to the Distributor program. This message arrives at the Distributor program and in turn causes the Distributor program to send a signal that it is ready to receive data.

At this point, an infinite loop is entered. At the beginning of the loop, the program waits to receive the joint angles of the manipulator, its end-effector position, and its wrist centre position from the Distributor Program. The end-effector position and wrist centre position can be calculated using the joint angles. However, the Manipulator Program has predefined functions to do such things, therefore, this option was used instead. At this point, the Controller Program also waits to receive joystick position and orientation data from the Joystick Program.

Determining the movement of the mobile manipulator when it is in State 2 depends on the knowing the last end-effector direction with respect to the mobile base frame, before it enters State 2. The Controller Program, therefore, constantly keeps track of the end-effector direction. This is simply done by subtracting the current end-effector position from the end-effector position of the previous iteration. The result is a vector that represents the end-effector's current direction of motion.

The joystick's position is then passed through a deadband position filter. The deadband position filter determines whether the centre of the triangular platform of the joystick is located within a virtual box as illustrated in Figure 4.9. The deadband position filter makes it easier for the operator to stop the mobile manipulator system, allowing him/her to be less precise. The process of determining whether the joystick

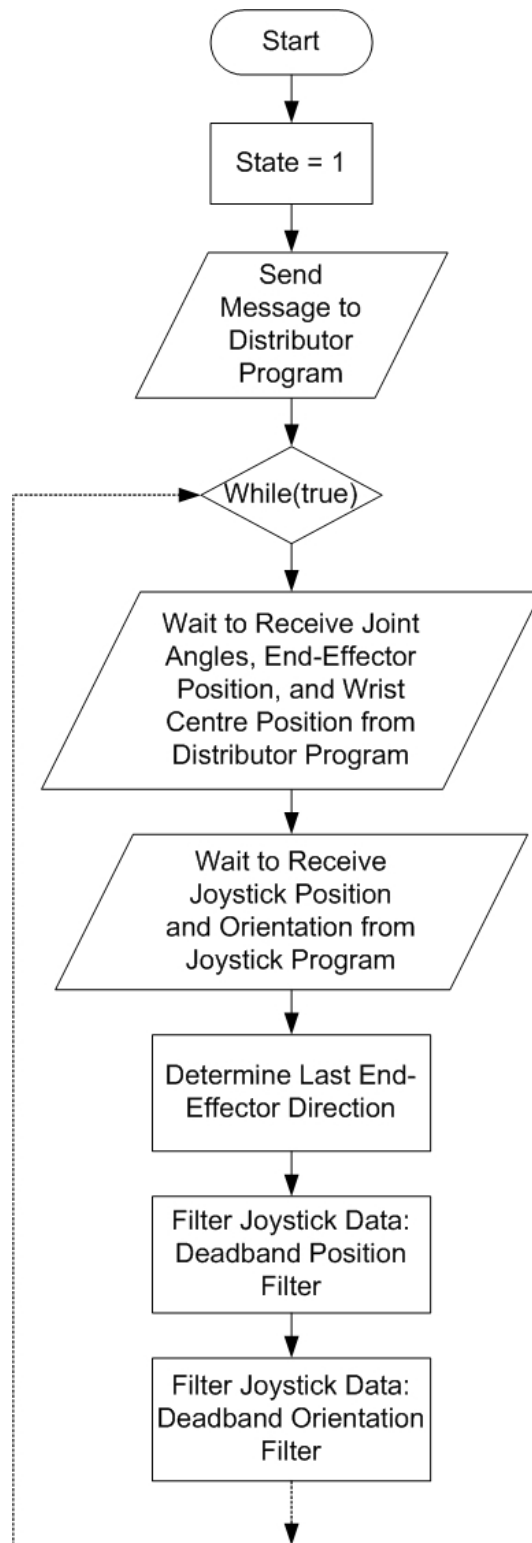


Figure 5.4: Controller Program Overview

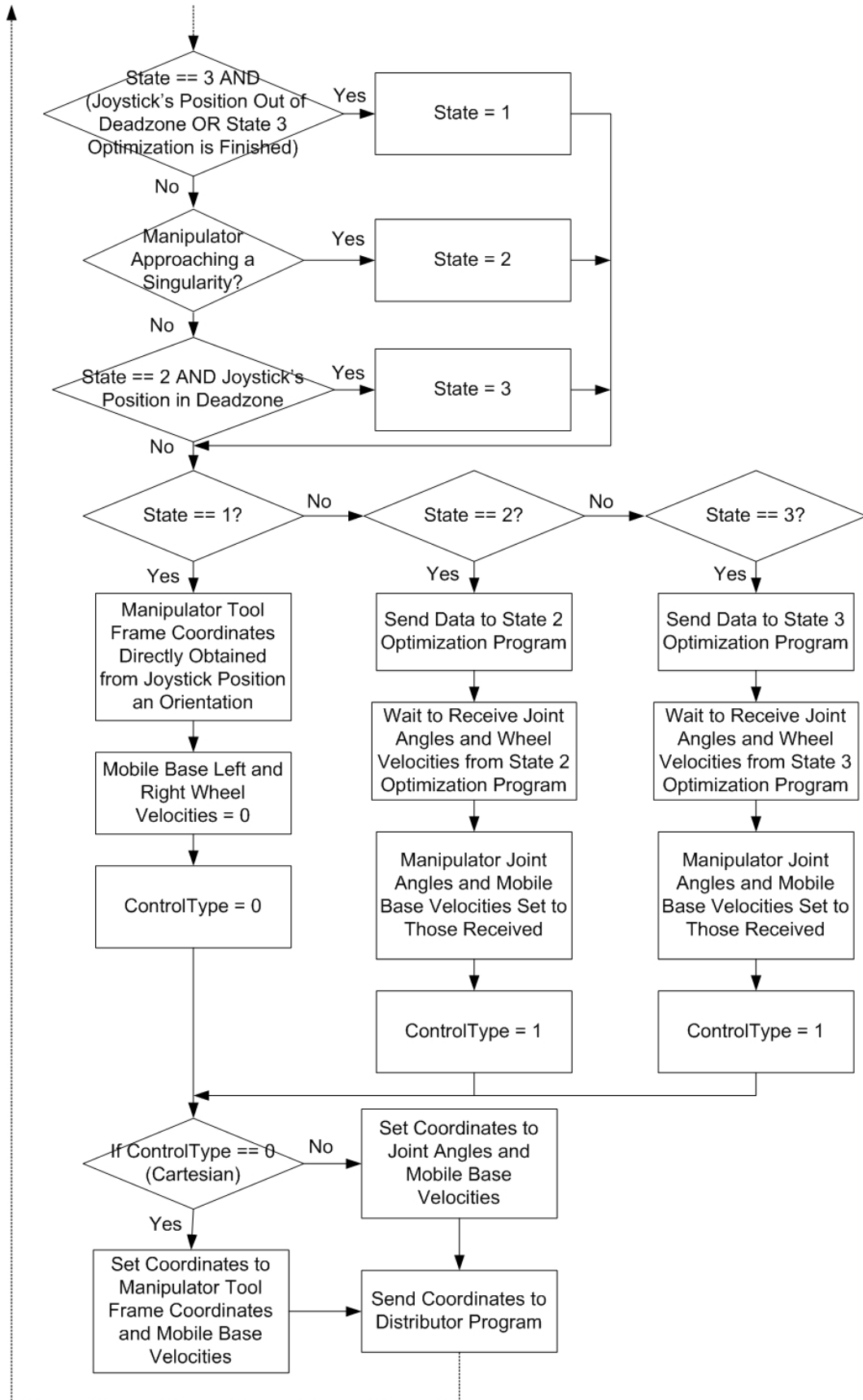


Figure 5.5: Controller Program Overview Cont'd.
105

is within this virtual box is illustrated in Figure 5.6.

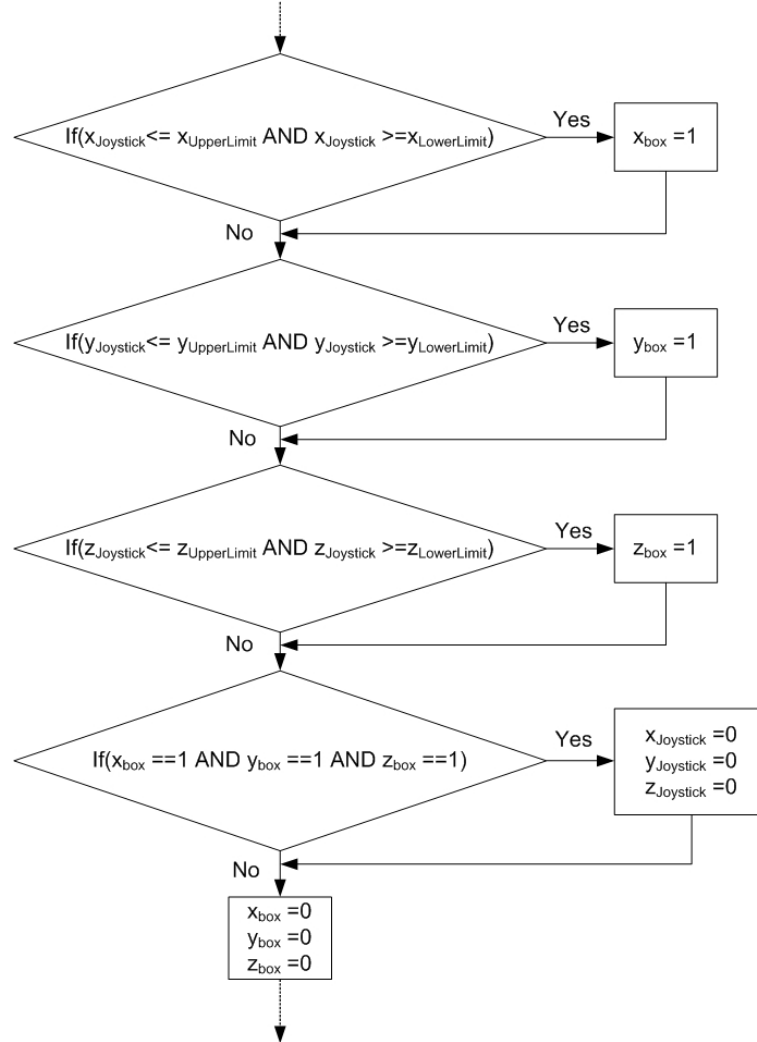


Figure 5.6: Position Deadband Filter Overview

The filter consists of three *if conditions*. The first *if condition* checks to see if the x component of the joystick position is between the upper and lower limits of the deadband filter, $x_{UpperLimit}$ and $x_{LowerLimit}$, respectively. If the x component is within this range, a flag variable x_{box} is set to 1 to indicate this. The program then enters the next two *if conditions*. They are identical except they check to see if the y and z components of the joystick position are between the upper and lower limits of the deadband filter, $y_{UpperLimit}$ and $y_{LowerLimit}$ and $z_{UpperLimit}$ and $z_{LowerLimit}$, respectively. If they are, then y_{box} and z_{box} are set to 1. If x_{box} , y_{box} , and z_{box} are all 1 then that

indicates that the joystick is within this deadband box and the joystick's position should be set to zero. During implementation, $x_{UpperLimit}$, $y_{UpperLimit}$, and $z_{UpperLimit}$ were set to 10 *mm* and $x_{LowerLimit}$, $y_{LowerLimit}$, and $z_{LowerLimit}$ were set to -10 *mm*. Likewise, the joystick's orientation is then passed through a deadband orientation filter.

The program then proceeds to decide what state of the control algorithm the mobile-manipulator system is in. As mentioned above, the states occur sequentially (State 1 is followed by State 2, State 2 is followed by State 3, and State 3 is followed by State 1 and so on). If the system is in State 1 as it is at the beginning, the only state that it can enter is State 2. It does this once the manipulator approaches a singularity. Setting *State* to 2 signifies this. The only way to exit State 2 and enter State 3 is through the second *if statement* and this only occurs if the joystick's position has been brought into the deadband, as described above. Lastly, once in State 3, there are two possible ways the system will re-enter State 1: if the joystick is pushed out of the position deadband or if the State 3 Optimization program finishes. This condition is implemented using the first *if statement*.

If *State* is equal to 1, the tool frame coordinates of the manipulator are set directly to those of the joystick. The left and right wheel velocities of the mobile base are set to zero, as in State 1, the mobile base is stationary. *ControlType*, another flag variable, is set to 0. This flag dictates what type of control is being employed. If *ControlType* is 0, incremental tool frame coordinates are to be sent to the manipulator. If *ControlType* is 1, joint angles are to be sent to the manipulator instead.

If *State* is equal to 2 or 3, the Controller Program sends data to either the State 2 Optimization Program or State 3 Optimization Program, respectively. It then waits for these programs to send back the necessary joint angles and wheel velocities. Once received, the manipulator joint angles and mobile base velocities are set and *ControlType* is set to 1, indicating joint control.

The calculated coordinates, along with the *ControlType*, are then sent to the Distributor Program.

5.5 State 2/State 3 Optimization Programs

The State 2 Optimization Program and State 3 Optimization Program are responsible for determining the mobile manipulator's motion during State 2 and State 3 of the algorithm, respectively. These two programs are very similar to one another as they are both based off of the DONLP2 optimization program [46]. Both are coded in ANSI C.

Allowing the motion during State 2 and State 3 to be determined by a program other than the Controller Program was done for a number of reasons. The base DONLP2 optimization program is very large, therefore, incorporating it into the Controller Program would be very problematic. Additionally, the DONLP2 optimization program conforms to the ANSI C standard and the Controller Program does not, which would mean restructuring one of the programs would be necessary. Moreover, by keeping the State 2 and State 3 control in separate programs, the system becomes more modular. If, for example, the method of control during State 2 needs to be changed/modified, then the State 2 Optimization Program is just replaced with little to no modification of the Controller Program. This further simplifies the system.

The structure and implementation of the two programs is identical although their functions are slightly different. The implementation is illustrated in Figure 5.7.

The State 2 and State 3 Optimization Programs begin by entering an infinite loop. The programs then wait to receive data from the Controller Program. The Controller Program only sends data to either one of these programs if the mobile manipulator is in State 2 or State 3, respectively. This data contains information on the current end-effector and wrist centre position as well as orientation. After which, the programs

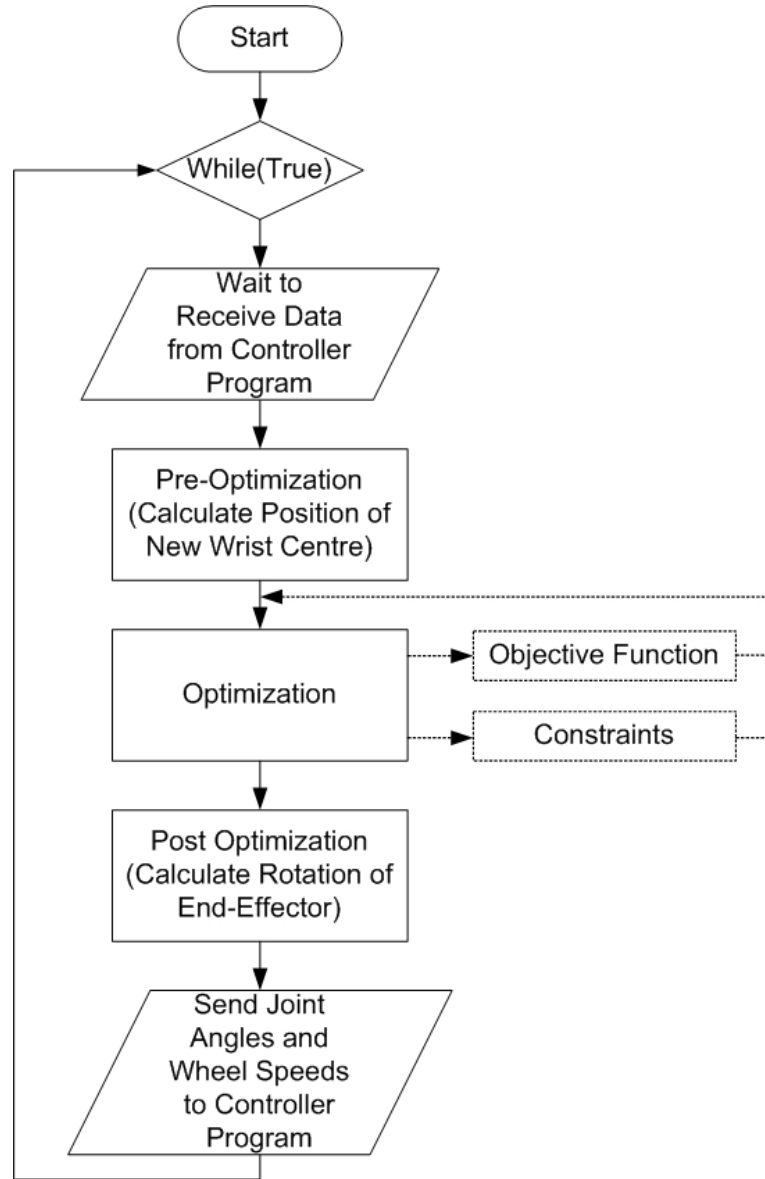


Figure 5.7: State 2 and State 3 Optimization Program Overview

enter the Pre-Optimization stage. In the Pre-Optimization stage, the new wrist centre position is calculated. The steps for this vary based on the program and are discussed in Sections 4.4.1 and 4.5.1 for State 2 and State 3 , respectively. The Optimization stage occurs next. The joint angles of the manipulator and wheel speeds of the mobile base are varied in order to minimize an objective function. This is all done while satisfying constraints that govern wrist centre movement, joint ranges, and the nonholonomic nature of the wheeled base. The objective function places the wrist

centre at the position calculated in the Pre-Optimization stage. The full details of the Optimization are discussed in Sections 4.4.2 and 4.5.2 for State 2 and State 3, respectively. Lastly, the program enters the Post-Optimization stage. Here, the end-effector's orientation is calculated in order to keep it in its current orientation with respect to the world frame. The full details of the calculations to do so are discussed in Sections 4.4.3 and 4.5.3 for State 2 and State 3, respectively. After this, the joint angles and wheel velocities of the mobile base are sent to the Controller Program. The program then loops back to the top.

It should be noted that State 2 only ends when the joystick is pulled back into the deadband region. State 3 ends when either the joystick is pushed out of the deadband or when the State 3 objective function is minimized past a certain user specified value. This value defines how close the program should bring the mobile manipulator to the ideal configuration before stopping.

5.6 Distributor Program

The Distributor program is responsible for two things: directing communication between the Controller Program and the manipulator and mobile base and keeping the manipulator and mobile base synchronized with one another. The Distributor Program has four main functions: it receives data from the Manipulator Program, it sends this data to the Controller Program, it receives coordinates from the Controller Program, and it sends these coordinates to the Manipulator and Mobile Base Programs.

The Distributor Program was written in C. It communicates with the Controller Program using QNX's message passing routines. It communicates with the Manipulator Program through the use of a null modem cable and with the Mobile Base Program through the use of wireless Ethernet. It runs on the QNX computer.

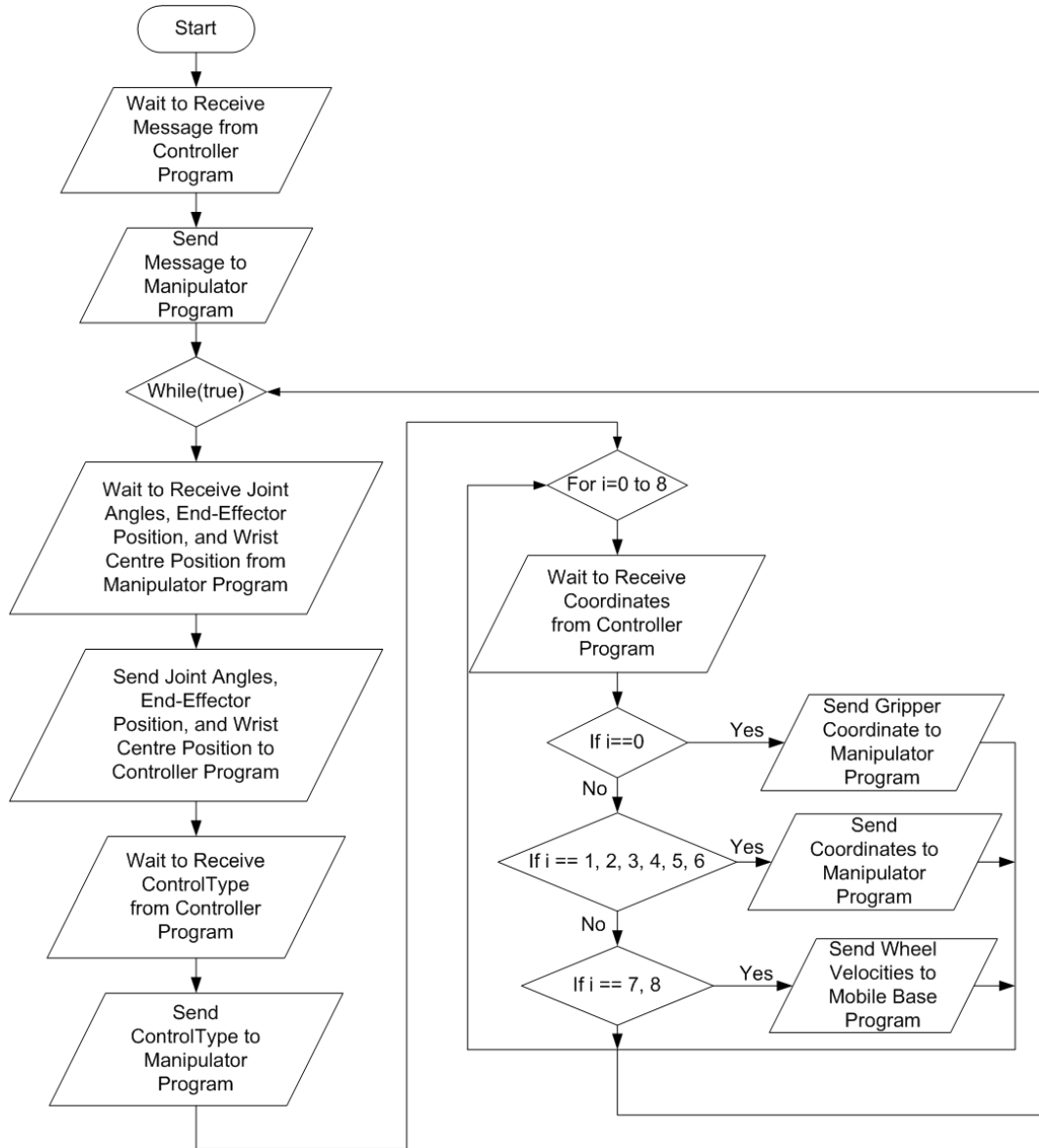


Figure 5.8: Distributor Program Overview

The function of the Distributor Program is illustrated in Figure 5.8. Initially, the Distributor Program waits to receive a message from the Controller Program. This message indicates that Controller Program is ready to receive data from the Manipulator Program. Therefore, a message is then sent to the Manipulator Program. This signals the Manipulator Program to send data back. If this did not occur, there is the possibility that either the Manipulator Program would send data to the Distributor Program without the Distributor Program being ready to read the data or

that the Distributor Program would send data to the Controller Program without the Controller Program being ready to read the data, resulting in data being lost.

The Distributor Program then enters an infinite loop. It first waits to receive the manipulator's joint angles, the end-effector's position, and the wrist centre's position from the Manipulator Program. The program then sends this data to the Controller Program as feedback. Based on this feedback and the joystick's movement, the Controller Program determines the motion of the mobile manipulator. Therefore, the program now waits to receive *ControlType* from the Controller Program. Again, *ControlType* will determine whether the Controller Program is sending incremental position and orientation movements or joint angles. *ControlType* does not affect the operation of this program. Once the variable is received, it is then sent to the Manipulator Program.

The program then enters a *for loop* that loops for a total of nine iterations (incrementing i from 0 to 8). At the start of each iteration, the program waits to receive coordinates from the Controller Program. The first coordinate is the distance between the gripper's fingers, this is sent to the Manipulator Program. The next six coordinates that are received are either the position and orientation movements or joint angles, these are sent to the Manipulator Program one by one as well. The last two coordinates received are the desired wheel speeds of the mobile base. Once received, these are sent to the mobile base. The program then loops back to the start of the infinite loop and repeats itself.

5.7 Manipulator Program

The Manipulator Program is directly connected to the manipulator and issues its motion commands. It has five basic functions: it prepares the manipulator for teleoperation, it sends manipulator information to the Distributor Program, it receives

motion coordinates from the Distributor Program, it sends motion commands to move the manipulator accordingly, and it attempts to smooth the motion of the manipulator.

The program runs on the F3 C500C controller. Since the C500C controller has no real visual output, a Console Computer is connected to it via a serial cable. The Console Computer allows the operator to see what is happening on the C500C controller. It runs Robcomm3, an integrated development environment (IDE), for the F3 articulated arm. This is where the Manipulator Program is developed, compiled, and uploaded to the controller. The Manipulator Program is written in the RAPL-3 robot programming language which is very similar to the C programming language. Once the program is uploaded to the controller, the Console Computer is used to start it. The output of the program is then displayed on the Console Computer.

The Manipulator Program begins by setting options for the manipulator's control. It specifies three things. It first specifies that all operations will be performed using metric units, millimeters to be exact. It then specifies the speed of the manipulator. During testing, the manipulator's speed was set to 10, defined as 10% of the maximum speed. Setting the speed very high is not advisable. The higher the speed, the less time the operator has to react. Lastly, *online mode* is enabled. This designates a queue in the C500C controller with space for up to eight motions to be stored. However, only two motions are stored in the queue at a time, the motion that the manipulator is in the process of currently performing and the upcoming desired motion. The purpose of the queue is to make sure motion of the manipulator is smooth. If there is always a set of motion coordinates in the manipulator's queue, the manipulator will not have to stop after it completes its current motion, it will already know where to go next. In order for the queue to be implemented though, control of the end-effector needs to be one iteration behind. This is not a major issue as motions are relatively small. If *online mode* is disabled, the queue only has space for one motion, the current

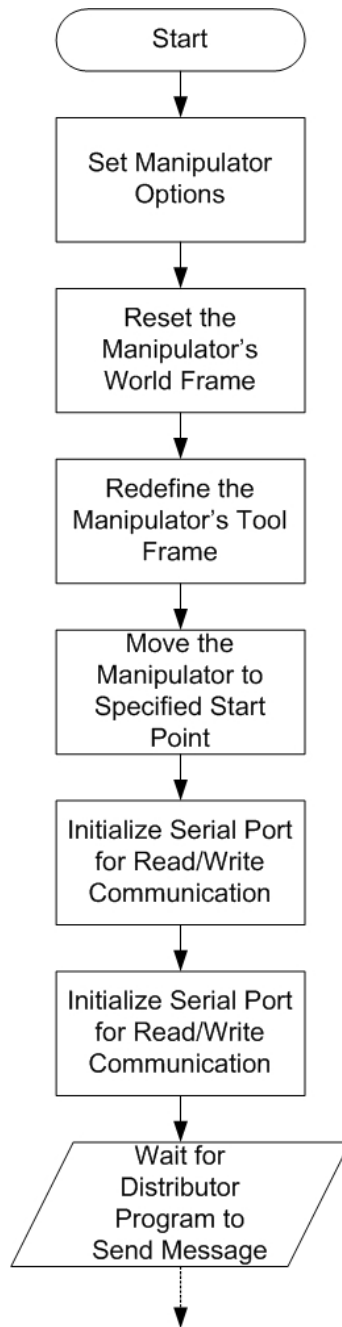


Figure 5.9: Manipulator Program Overview

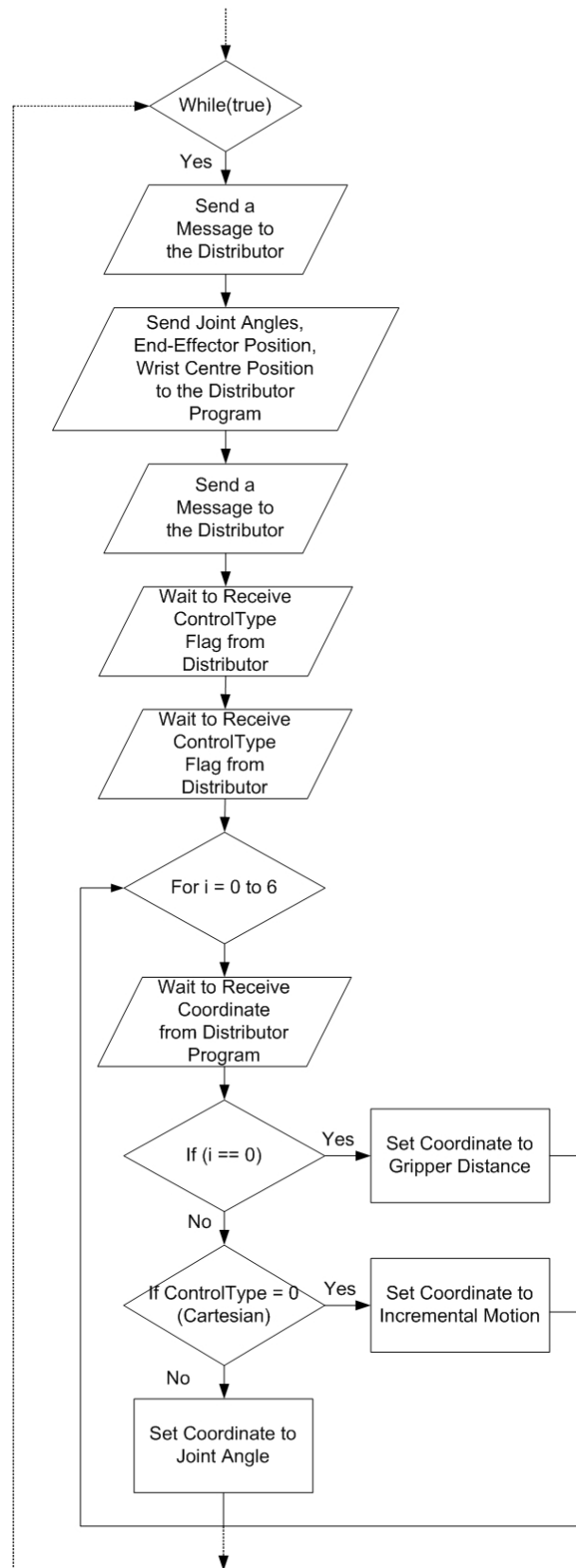


Figure 5.10: Manipulator Program Overview Cont'd.

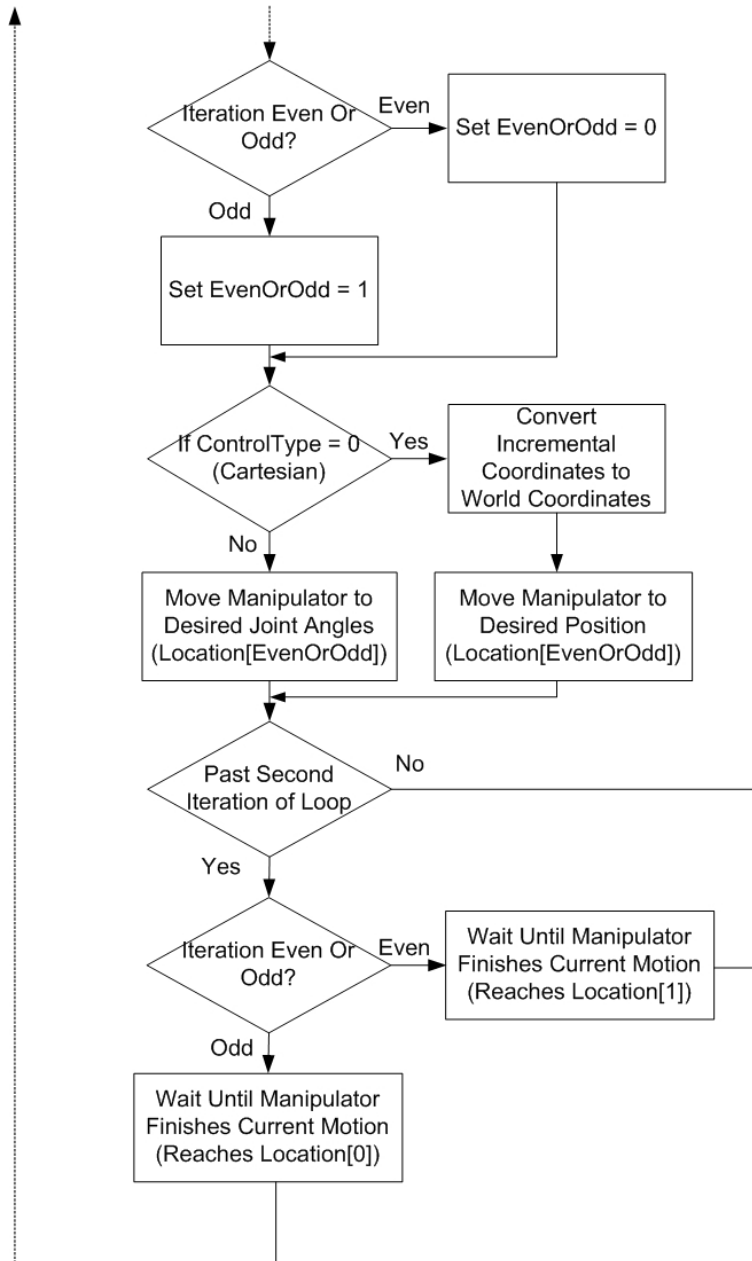


Figure 5.11: Manipulator Program Overview Cont'd.

motion. When this motion is completed, the queue is only then ready for a new motion to be put in. Therefore, the manipulator will have to stop for a short period of time before it proceeds again. The manipulator's motion would exhibit extreme jerkiness.

The manipulator's world frame is then reset and the tool frame is redefined. The default parameters of the manipulator are loaded when the C500C controller is restarted. When a program runs on the C500C controller and changes parameters such as the location of world frame or tool frame, these changes stay in effect even after the program is finished. Therefore, it is important to redefine these parameters at the start of the program. The manipulator's world frame is set right in the middle of its base. The tool frame is set at the tip of the end-effector. The default tool frame assumes that the F3 articulated arm uses the manufacturer's gripper fingers.

The manipulator is then moved to a specified start point. This start point coincides with the ideal pose, a manipulator configuration that has high manipulability. It ensures that the manipulator's end-effector is in a good position once control starts. For example, it would not be ideal for control to start with the manipulator in a singular condition. The Manipulator Program waits until the manipulator moves into this start position and then proceeds.

At this point the serial port is initialized for read/write communication. It is configured with the following properties: a baud rate of 57,600 bps, an 8 bit word length, no parity, and 1 stop bit.

The Manipulator Program now waits for a message from the Distributor Program using what is known as a *blocking read* function. A *blocking read* function waits for a message to be received as opposed to a *non-blocking read* function that does not and just proceeds with the subsequent instructions. The message from the Distributor Program indicates that the Distributor Program is ready to read information that is to be sent. The purpose of waiting is just a precaution in case the Distributor program

is not yet ready to read, but the Manipulator Program is ready to send messages. Messages from the Manipulator Program are sent to the Distributor program using a *non-blocking write* function (i.e., it proceeds regardless if the receiving program has received the messages). If the messages were sent before the Distributor Program was ready to read, then they simply would be lost and the Distributor Program would wait for the next messages to come. However, since no messages are sent afterwards, the Distributor Program would wait forever.

The program now enters an infinite loop, until the operator decides to terminate the program. The current end-effector location is retrieved. The location is converted into the current joint angles using a RAPL-3 function that is based on the inverse displacement solution. The six joint angles, the position of the tool centre, and the position of the wrist centre are all sent to the Distributor Program. Again, they are sent using a *non-blocking write* function. From the Distributor Program this data is sent to the Controller Program as part of the feedback of the system.

Since there is no direct function to find the wrist centre of the manipulator a number of steps need to be taken to do so. The current tool centre's position is retrieved as a *cloc*, *ToolCentre*. A *cloc* is a RAPL-3 data type that “represents a point in the robot arm workspace defined in cartesian co-ordinates” [48]. Within each *cloc*, the three position coordinates, x , y , z and three orientation components, α , β , γ are specified. The coordinates of *ToolCentre* will be specified as: x_{TC} , y_{TC} , z_{TC} , α_{TC} , β_{TC} , and γ_{TC} . Using, the three orientation components, α_{TC} , β_{TC} , and γ_{TC} , and assuming x_{TC} , y_{TC} , z_{TC} are all zero, a new *cloc*, *OrientationTool*, is created. This *cloc* represents the orientation of the tool frame located at the world frame. A function that “alters the cartesian coordinates of a location in the tool frame of reference” [48] is now used to shift *OrientationTool* by *ToolOffset*, another *cloc* that represents the wrist centre's position with respect to the tool centre in the tool centre frame. This results in *WristOffset*, a *cloc* that represents the wrist centre's position with respect to the

tool centre in the world frame. If x_{WO} , y_{WO} , z_{WO} , α_{WO} , β_{WO} , and γ_{WO} represent the coordinates of *WristOffset*, then the coordinates of *WristCentre*, a *cloc* that specifies the location of the wrist centre, can simply be found as follows:

$$x_{WC} = x_{TC} + x_{WO} \quad (5.1)$$

$$y_{WC} = y_{TC} + y_{WO} \quad (5.2)$$

$$z_{WC} = z_{TC} + z_{WO} \quad (5.3)$$

$$\alpha_{WC} = \alpha_{TC} \quad (5.4)$$

$$\beta_{WC} = \beta_{TC} \quad (5.5)$$

$$\gamma_{WC} = \gamma_{TC} \quad (5.6)$$

Now the Manipulator Program is ready to receive motion coordinates from the Distributor. The program waits to receive *ControlType*, a flag variable that indicates what type of control is needed. As described in Section 5.4, when the manipulator is either in State 2 or State 3, direct joint control is used as some joints need to be fixed. If *ControlType* is 0, then position coordinates will be sent by the Distributor Program. If *ControlType* is 1, then joint angles (motor pulses) will be sent.

The program now enters a *for loop* that loops for eight iterations, for the eight coordinates that are to be received. During the start of each iteration, the program waits to receive each coordinate. The first coordinate deals with the gripper, specifically the distance between the gripper's fingers. This coordinate increments the current distance between the gripper's fingers. If this incremental coordinate causes the gripper to exceed its maximum range of 5.08 *cm* then the gripper is set to its maximum value. If the incremental coordinate causes the distance between the gripper's fingers to be less than zero, then the gripper is set to zero.

The next six received coordinates are assigned to one of two arrays, *Incremental*[] or *Joint*{}. If *ControlType* is 0, then position coordinates are currently being received

and therefore are assigned to the six elements of the *Incremental*[] array. If, however, *ControlType* is 1, then joint pulses are being received and are assigned to the six elements of the *Joint*[] array.

The program then checks to see if the current iteration of the infinite loop is even or odd. This is important in dealing with the queue. To determine this, the following is used:

$$EvenOrOdd = Iteration \bmod 2 \quad (5.7)$$

The modulo (mod) operation determines the remainder when *Iteration* is divided by 2. Since the remainder of any even number divided by 2 is zero, *EvenOrOdd* is 0 if the iteration is even and 1 if the iteration is odd.

Now that the new coordinates are known, the program determines the next location that the manipulator must move to. Again, this depends on the type of control that is desired.

If position control is being used, a few steps are required in transforming the current received coordinates (which are defined with respect to the tool frame) to the world frame. This transformation is necessary as there are no RAPL-3 functions that directly control the end-effector with respect to the tool frame using all 6-DOF. The only functions that exist in this respect, *jog* the manipulator in one direction or in one orientation (with respect to the tool frame) at a time. Transformation functions do exist in the RAPL-3 language, however, and are used. First, a *cloc*, denoted as *Incremental*, is built using the incremental coordinates (those of the *Incremental*[] array that have been received). Another *cloc*, *World*, is then constructed using the current world coordinates of the end-effector with respect to the world frame. Using a RAPL-3 function, the *Incremental cloc* is now shifted by the *World cloc* with respect to the world frame. This results in the *cloc*, *IncrementedWorld*. The orientation components of *IncrementedWorld* represent the orientation of the desired end-effector position with respect to the world frame.

To find the translational component required, two *clocs* are created one called *Orientation* and the other called *Translation*. *Orientation* has no translational components and consists purely of the three orientation components from the *World cloc*. *Translation* has no orientation components and consists purely of three translational components, those of *Incremental*. *Orientation* is then shifted by *Translation* with respect to the tool frame. The resulting *cloc*'s translation components are then used in determining the position change with respect to the world frame.

A *cloc*, *loc[EvenOrOdd]* is then created using these desired end-effector position and orientation components (that are defined with respect to the manipulator's world coordinate system). The manipulator is then instructed to move to this position and orientation. Since the manipulator is currently completing a task, the position and orientation go into the queue.

If joint control is being used, moving the joints to their desired location is much simpler. In this case, a *ploc* is created using the six joint angles found in the *Joint[]* array. A *ploc*, *JointMotion*, is a RAPL-3 data type that "represents a point in the robot arm workspace defined by increments of rotational movement, specifically encoder counts, of each joint of the arm" [48]. Within each *ploc*, all six joint positions are defined. The Manipulator Program then simply tells the manipulator to move to the location described by *JointMotion*. The *JointMotion cloc* is then converted to world coordinates and set to *location[EvenOrOdd]*.

Now if the loop is past its first iteration, at this point in the program, the program checks to see if the first motion in the queue has been completed. This is only done after the first iteration because that will ensure two motion commands are in the queue. The first motion command is always the one currently being executed by the manipulator and the second is the upcoming motion. Since the program waits for the first motion to be completed before proceeding, this also ensures that no more than two motion commands are in the buffer. To achieve this, the program enters an *if*

condition (if the current iteration is past the first).

If the current iteration is even, the program checks to see if the manipulator has reached *location*[1], the odd location. The odd location is the motion that is currently being completed in this case. If this location is not yet reached, the manipulator waits until it does. This is done through the use of a *while loop*. The *while loop* checks to see whether any one of the three translation components are within a specified distance, *postol*, and whether any one of the three orientation components are within a specified angle, *orienttol*, from their desired position. If even one of the components is not within its desired position, the program enters the loop, retrieves the current position, and again checks the condition. The program exits the loop only if the current position is within the tolerance of *location*[1]. If the current iteration is odd, the program checks to see if the manipulator has reached *location*[0], the even location, instead. The even location being the motion that is currently being completed.

Waiting for the manipulator to finish the motion that it is currently completing ensures that the manipulator program does not run too fast. This makes sure only two motions are in the queue, the current motion of the manipulator and the next motion of the manipulator. If the manipulator program executed too quickly, it is possible that the queue would fill up very fast and that there would be a significant lag between the joystick's position and orientation and the manipulator's position and orientation.

5.8 Mobile Base Program

The Mobile Base Program is the program that runs on the PowerBot's onboard computer and directly controls its motion. It has four main functions: it configures the mobile base for tele-operation, receives wheel velocities from the Distributor program, instructs the mobile base to move, and sends information back to the Distributor.

The program is written in C++. It uses the ActivMedia Robotics Interface for Ap-

plications (ARIA) software. ARIA contains a large amount of functions that allow various aspects of the PowerBot AGV to be controlled. The ARIA functions communicate directly with the PowerBot's hardware controller making it very simple to physically control the mobile base and to understand the Mobile Base Program.

The Console Computer is used to wirelessly start the program. A program such as Telnet or SSH is used to connect to the PowerBot's onboard computer over the wireless Ethernet network. These two programs are able to view the entire system directory of the computer and are able to start the Mobile Base Program.

The structure of the Mobile Base program is illustrated in Figure 5.12. The program starts by initializing the connection handler. The connection handler is responsible for a number of things. It first connects the Mobile Base Program directly to the hardware controller. If it successfully connects, then the sonar panels and sounds are turned off and the motors are turned on. If the connection fails or is lost for some reason, the connection handler immediately exits the program.

The socket server is then started. The socket server allows for the Distributor Program to connect through the use of wireless Ethernet. The PowerBot has been configured to acquire a static IP address every time it is started. The IP address is always the same and there is no way any other computer can claim this IP address as the system is on a dedicated network. This is done so that the Distributor Program will always be able to connect to it. The socket server is opened on an unused specified port. Two potential types of communication protocols for the socket server exist: Transmission Control Protocol (TCP) and User Datagram Protocol. TCP guarantees that data is delivered and is correct, while UDP does not, however, UDP operates at a higher speed. As reliability is very important in the control of the mobile manipulator system, TCP was chosen as the protocol of choice. Once the socket server is set up, the program waits for the Distributor Program to connect.

The program then enters an infinite loop. It then waits a set period of time to receive

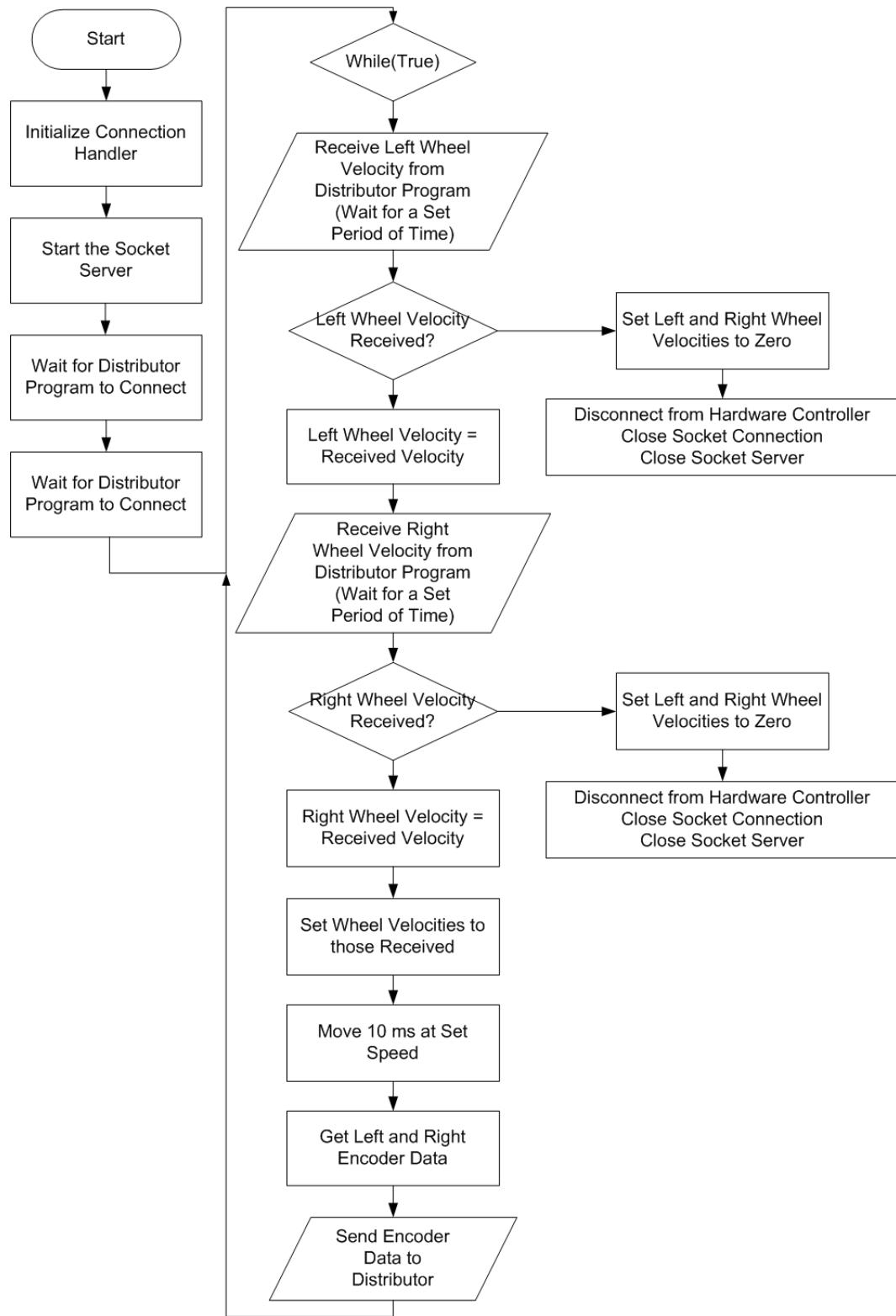


Figure 5.12: Mobile Base Program Overview

the left wheel velocity from the Distributor Program. The predefined socket read function is nonblocking. It does allow the read function to wait for a signal for a set period of time. This however does not pose a problem. Rather than using a large time, a time of 5 seconds was specified during testing. It is assumed that if data does not arrive within this period of time, something must be wrong with the system. The program then waits to receive the right wheel velocity from the Distributor Program, again, for a set period of time. If the left or right wheel velocities are not received, the wheel velocities are set to zero, which ensures that the mobile base safely stops. The program is also disconnected from the hardware controller, and both the socket connection and socket server are closed. If the socket connection and socket server are not closed, starting the Mobile Base Program again may cause a problem as the program will not be able to connect to a port that is currently open.

If there are no problems, the left and right wheel speeds will be set to those received. The mobile base will then move for 10 ms at the set speed. The left and right wheel encoder counts are then retrieved from the hardware controller. They are then sent to the Distributor program and back to the Controller. Currently, they are used for monitoring purposes. It should be noted that a velocity controller is built into the function that is used to control the velocity. This keeps the velocity as close to the desired velocity as possible. After this, the program loops back to the top of the infinite loop.

5.9 Summary

The novel control algorithm has been implemented on the Jasper mobile-manipulator system. The implemented system consists of a total of seven programs that run simultaneously. The programs have been created to be modular and keep the system easy to understand, rework, and update.

The Joystick Program reads encoder data from the 6-DOF joystick. Using this data, it first calibrates the joystick. After which, it uses the joystick's forward displacement solution to find its position and orientation. These coordinates are then sent to the Controller Program. This program decides what state the system is in and makes movement decisions accordingly. If the mobile manipulator is in State 2 or State 3, the Controller Program calls upon the State 2 and State 3 Optimization Programs to solve for the necessary motion, respectively. The Controller Program sends the end-effector coordinates to the Distributor Program. This program is responsible for communication between the Controller Program and the manipulator and mobile base. It also synchronizes the manipulator and mobile base. These five programs run on the Operations Computer.

The Manipulator Program runs on the manipulator's controller. It receives coordinates from the Distributor Program and instructs the manipulator to move. It then sends feedback back to the Distributor Program.

The Mobile Base Program runs on the mobile base's computer. It receives wheel velocities from the Distributor Program. It then instructs the mobile base to move at those velocities.

Chapter 6

Results and Discussion

Successful test results have been obtained in implementing the proposed control algorithm. The results validate the control algorithm in two ways. First, the results show that the proposed control algorithm can be implemented on hardware. It is not just a theoretical idea, but a proven one. Secondly and most importantly, the results demonstrate that the control algorithm serves as a feasible method of control, allowing the mobile manipulator to be controlled in a unified and intuitive manner.

Two types of tests have been conducted to verify the control algorithm: motion tests and an application test. The motion tests check to see if the control algorithm moves the mobile manipulator the way it should in States 1 to 3. The application test checks to see if the algorithm as a whole works. It shows the algorithm's effectiveness in a simulated real-world application.

6.1 Motion Tests

The motion tests were used to check the motion of the system and compare it with its theoretical motion.

6.1.1 State 1

State 1 occurs when the manipulator is far from singularities (i.e., the end-effector is well within the work envelope). During this state, the joystick directly controls the manipulator. The distance and orientation change of the joystick from its home frame $\{H\}$ represent the scaled incremental position coordinates that are sent to the manipulator. State 1 was tested in two ways: using simulated joystick data and using real-time joystick data.

Simulated joystick data was used to verify if the motion of the end-effector represented that of the desired motion. In these tests, one joystick coordinate was set while the rest were set to zero. These tests proved that the control algorithm successfully moved the end-effector with respect to the manipulator's tool frame.

In the second test of State 1, the joystick was used. The joystick was moved around its workspace and the end-effector motion was noted. The joystick's position and orientation translated directly to the end-effector's position and orientation as expected. The end-effector's motion came to a stop when the joystick entered the deadband region. Orientation changes of the joystick needed to be pronounced in order to register orientation changes of the manipulator. It must also be noted that in this test, lag was exhibited between the joystick and the end-effector. This is believed to be a result of always storing the future coordinate in the buffer at all times, making the mobile manipulator one iteration behind the joystick.

6.1.2 State 2

State 2 begins when the manipulator approaches a singularity. In State 2, the mobile-manipulator system is required to move in such a way as to keep the end-effector moving in the direction that it was moving before it reached a singularity while keeping the end-effector in a constant orientation.

Four tests were carried out for State 2. Each test involved starting the mobile-

manipulator system with the manipulator in a configuration of high manipulability. The manipulator's end-effector was then directed to move outwards at angles of 0° , 45° , and 90° with respect to the base. In addition, it was also directed to move inwards at an angle of 180° . This way the last end-effector direction before approaching a singularity was known. In each case, predefined simulated joystick data was used as the input to the system. This was done in order to maintain consistency and accurately know the last end-effector direction.

Two things were observed during each test: the way the algorithm monitored singularities and whether or not the end-effector continued to move in the direction that it was moving prior to reaching a singularity while maintaining its orientation.

Figures 6.1 and 6.2 show the manipulator moving forward from State 1 to State 2. The controller effectively initiates movement of the base once the arm approaches a singular configuration as indicated in the series of pictures. The controller successfully controls the manipulator's end-effector to keep moving in its last direction (forward direction).

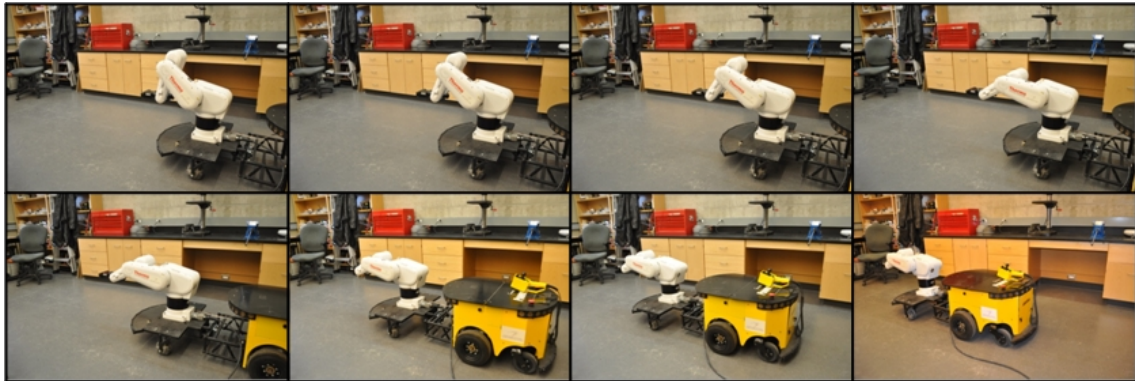


Figure 6.1: State 2: Last End-Effector Direction 0° (Side View)

In Figures 6.3 and 6.4 the manipulator moves at a 45° angle with respect to the base's orientation. Again, the controller effectively initiates movement of the base once the manipulator approaches a singularity. The end-effector then continues to move in the same direction. Both the base and joint 1 turn to keep the end-effector moving as

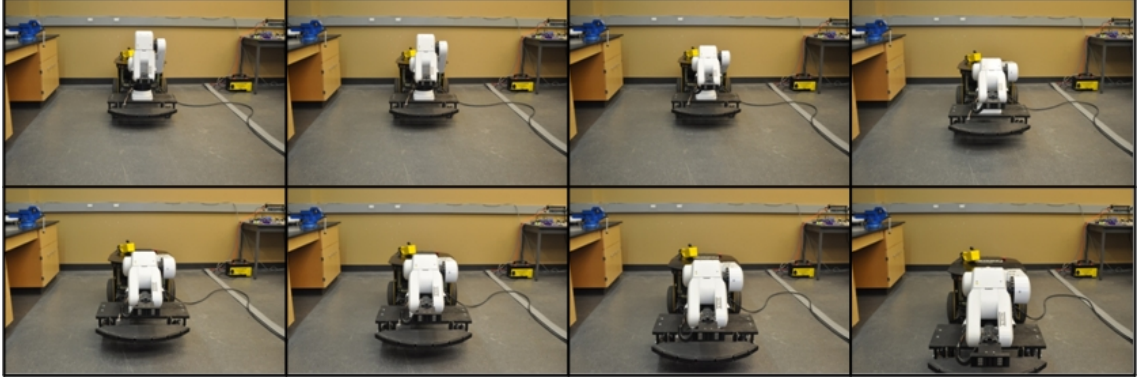


Figure 6.2: State 2: Last End-Effector Direction 0° (Front View)

desired. The orientation of the end-effector remains roughly constant.



Figure 6.3: State 2: Last End-Effector Direction 45° (Side View)

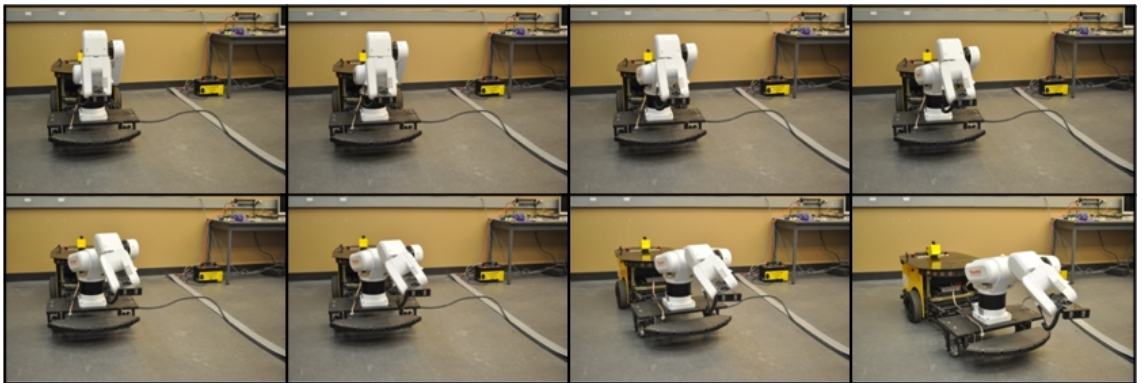


Figure 6.4: State 2: Last End-Effector Direction 45° (Front View)

The manipulator moves at a 90° angle (sideways) with respect to the base in Figures 6.5 and 6.6. Once the manipulator approaches a singular configuration, the mobile

base moves. As before, joint 1 turns to compensate for the change in orientation of the base. Again the end-effector continues to move in the direction that it did before with minor deviation, while roughly maintaining the end-effector's orientation.

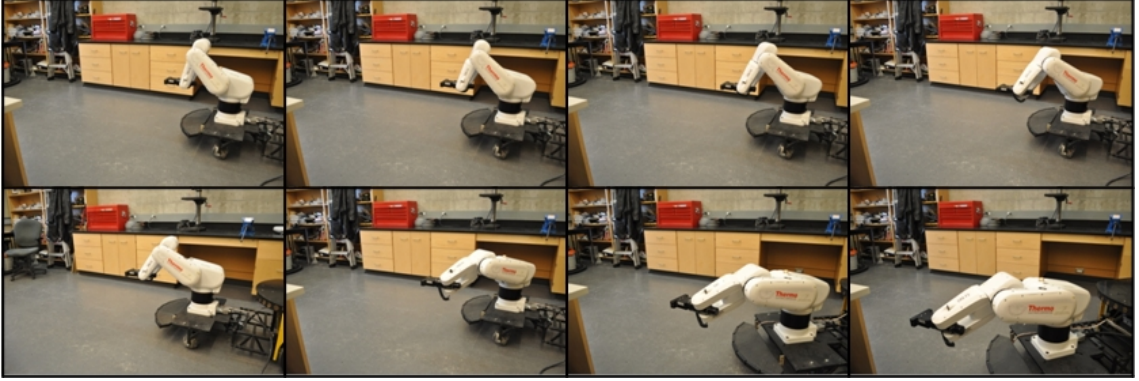


Figure 6.5: State 2: Last End-Effector Direction 90° (Side View)

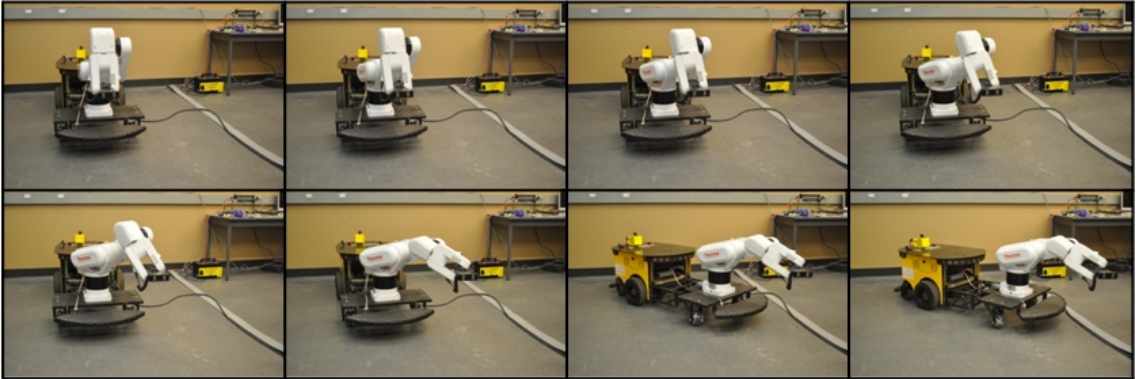


Figure 6.6: State 2: Last End-Effector Direction 90° (Front View)

Figure 6.7 shows the results of the manipulator moving inwards at 180° . When the manipulator approaches the singularity, the mobile base moves the manipulator backwards. This keeps the end-effector moving in the same direction as it did before it approached the singularity.

In all four instances, the control algorithm successfully detected the manipulator approaching a singularity. The end-effector then translated in the direction that it was traveling (prior to having approached the singularity) with minor error. In the 45° and 90° instances, the end-effector's orientation was roughly maintained.

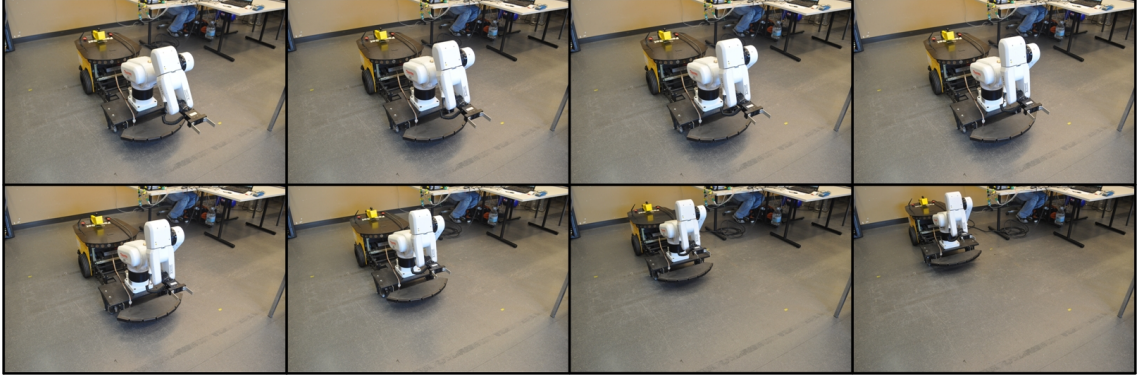


Figure 6.7: State 2: Last End-Effector Direction 180°

6.1.3 State 3

After State 2, the mobile manipulator enters State 3. In State 3, the end-effector's position and orientation remain constant with respect to the world frame as the manipulator and mobile base move in unison in order to bring the manipulator into a more ideal configuration.

The results of one of the State 3 tests are shown in Figure 6.8. The manipulator begins in a near-singular configuration, as it exits State 2. The end-effector is slightly offset to the side to better test the capability of the control algorithm (as this means the mobile base will have to rotate). Once initiated, the manipulator and mobile base move in unison. The end-effector's position and orientation does not remain perfectly constant (with respect to the world frame). The position is offset by a few centimetres and the orientation is offset by a few degrees. However, it is important to note that this offset occurs at the beginning of State 3. Once into State 3, there was very little position and orientation change of the end-effector noted.

The motion of the manipulator and mobile base in State 3 exhibits some jerkiness. This is believed to be either a result of the length of the optimization process or a fault in keeping one motion in the buffer at all times.



Figure 6.8: State 3: Results

6.2 Application Test

In order to test the overall effectiveness of the control algorithm, an application test was developed. Unlike the motion tests, which prove that the control algorithm controls the mobile-manipulator as it should in a prescribed manner, the application test checks to see how well the control algorithm works in practice.

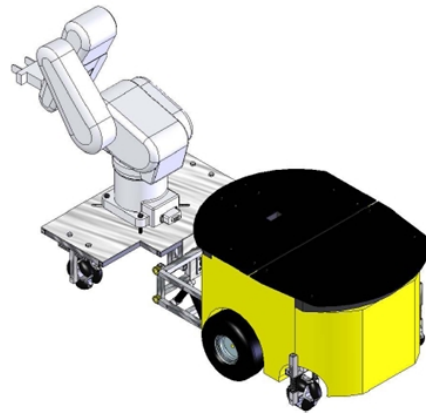
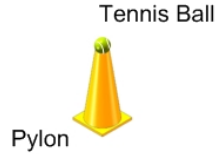


Figure 6.9: Application Test

The application test is used to simulate a real-world task of positioning the end-effector to do some desired task. The task begins with a tennis ball, located on a

pylon, placed at a distance in front of the mobile-manipulator system, as shown in Figure 6.9. The goal of the test is for the operator to control the mobile manipulator in such a fashion as to traverse the distance and pick up the tennis ball with the end-effector.

Figures 6.10 to 6.12 show the mobile manipulator being tele-operated. Figure 6.10 shows the mobile manipulator in State 1. Here, the operator pushes the joystick forward (towards the target) causing the end-effector to move forward and approach a singularity. Figure 6.11 shows the motion of the mobile manipulator once the manipulator approaches the singularity. The mobile base initiates movement, keeping the end-effector moving in its last direction prior to having approached a singularity. During this period, the manipulator remains outstretched and the joystick is still pushed forward. Lastly, Figure 6.12 shows the mobile manipulator in State 3. State 3 is initiated by the operator when the end-effector is directly over the tennis ball. The operator pulls the joystick in towards the deadband region. The manipulator then moves in conjunction with the mobile base in such a way as to keep the end-effector's position and orientation stationary with respect to the world frame. The end-effector is now in an ideal position to pick up the tennis ball.



Figure 6.10: Application Test Results: State 1

The results of the application test were very good. The mobile manipulator *sensed* that the operator required to move forward and did so. When the desired position was reached, the mobile manipulator then stopped at the request of the operator and moved into an ideal configuration for manipulation. Overall, the algorithm successfully allowed the operator to tele-operate the system in an intuitive manner.

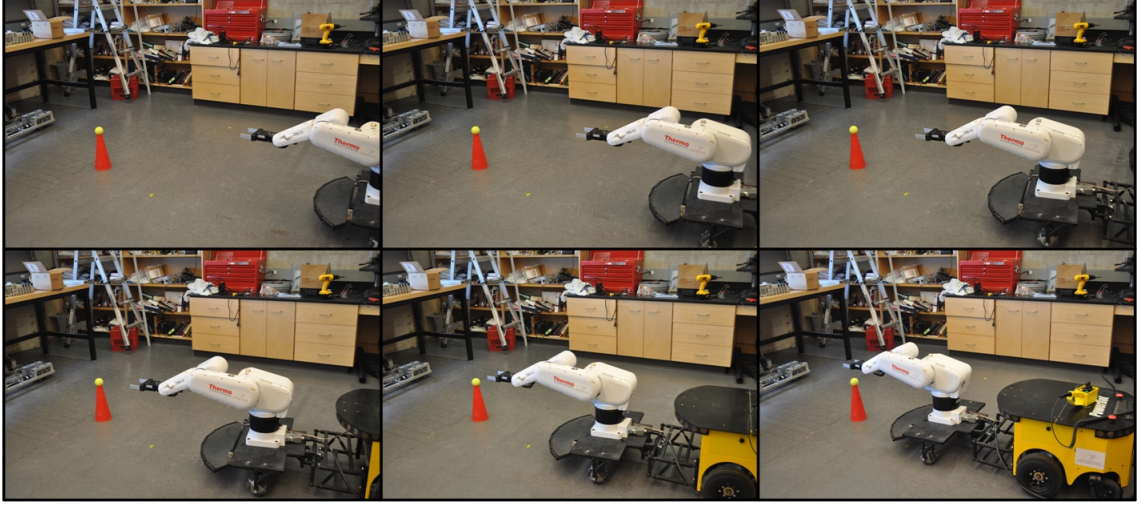


Figure 6.11: Application Test Results: State 2



Figure 6.12: Application Test Results: State 3

6.3 Sources of Error

There are several possible sources of error in the results. In State 2, while keeping the end-effector moving in its last direction, minor error in end-effector position and orientation was noted. In State 3, while the mobile manipulator reconfigured itself to an ideal pose the end-effector did not remain completely still. There was minor error in the end-effector position and larger error in the orientation. There are a number of possible sources for this error.

For one, there is some wheel slip between the base and the floor. The velocity of the mobile base is controlled to achieve a certain translation and rotation of the base. If the mobile base does not achieve this velocity during the current iteration of the algorithm then it will not achieve the desired positional and rotational change causing error in the end-effector's position and orientation. To avert this, wheels with higher

traction could be used. As well, adding mass to the mobile base might help increase the friction force.

Error is also caused by the inertia of the system. Again, the control algorithm requires a certain velocity of the mobile base in order to achieve a desired position. When the mobile base is stationary it has a large inertia, due to its large mass. Therefore, it is very difficult for it to achieve the desired velocity that the control algorithm requires during its first few iterations. Once the speed of the mobile manipulator increases, this effect should decrease. This effect is pronounced very clearly in the State 3 test shown above. From a stationary position, the mobile manipulator enters State 3. The error in the end-effector's position and orientation is most pronounced at the beginning of the algorithm. Once the mobile manipulator is up to speed, changes in its velocity are much more manageable to achieve by the mobile base's controller. To prevent this, the controller should be adjusted to give reduced velocities during its first few iterations while it overcomes the stationary inertia.

It should also be noted that various other items could result in error. For example, an improperly calibrated joystick or manipulator or uneven tire pressure in the mobile base can cause error in the results.

Chapter 7

Conclusions and Recommendations for Future Work

7.1 Conclusions

A novel algorithm for simplified tele-operation of mobile manipulator systems has been developed. The algorithm was tested on an 8-DOF mobile-manipulator system, but could easily be applied to other systems.

The control algorithm is best described through three states. In State 1, the operator solely controls the manipulator through the use of a 6-DOF joystick. The joystick's position and orientation change with respect to a predefined home frame constantly increment the end-effector's position and orientation, respectively. If the manipulator approaches a singularity the system enters State 2. In this state, the mobile base and manipulator move in such a way as to keep the end-effector moving in its last specified direction with respect to the horizontal plane. During this motion, the manipulator remains extended outwards. The operator then returns the joystick to the home position to specify that the desired end-effector position is reached. This initiates the start of State 3. Both the manipulator and base move in conjunction

with one another so as to arrange the manipulator in an ideal pose, while keeping the end-effector's position and orientation constant in the world coordinate frame.

A basic implementation of the algorithm has been achieved using the Jasper mobile-manipulator system. The system is implemented across seven different programs, each with its own specific task. It has been designed to be modular and easily adjustable to changes.

Extensive testing of the algorithm was undertaken on Jasper. Two types of tests were conducted: motion tests and application tests. The motion tests showed that the mobile manipulator moved in accordance with the descriptions of State 1, State 2, and State 3. Small error in the end-effector's position and orientation were visible in both State 2 and State 3. This error can be attributed to wheel slippage and the mobile base's inertia. The system also exhibited some lag, however, this was expected. The application test simulated a real-world usage of the mobile manipulator system. The results were successful and showed that the developed control algorithm is intuitive and easy to use. It also allows the mobile manipulator to avoid singularities (both physical and algorithmic). The algorithm focuses on the operator and only uses one input device.

7.2 Recommendations for Future Work

Jasper is the very first mobile manipulator at the MARS Lab at UOIT. The presented novel control algorithm is also the very first of its kind. Consequently, there are still some additions that need to be made to the algorithm. They are listed as follows:

- **Refinement of State 2 Algorithm:** The proposed method of control during State 2 worked successfully. The only problem occurs if the operator wants to make a correction to the direction of travel during operation. Currently, the operator would have to enter State 3 to reconfigure the mobile manipulator then

State 1 to extend the manipulator in a new direction. A method to simplify this is required. It is suggested that the joystick be able to control the system's direction of travel during State 2.

- **Refinement of State 3 Algorithm:** The implementation of State 3 on the Jasper mobile-manipulator system is somewhat jerky. The problem is believed to be either in part to a fault in the coordinate buffer or the speed of the optimization routine. In any case, further research needs to be conducted to smooth the motion.
- **Obstacle avoidance:** Obstacle avoidance should be added to the control algorithm. There are situations where obstructions may be in Jasper's path, for example if there is an obstruction in front of the mobile base and the user enters State 3, it could cause the base to hit the obstruction. The bump sensors and existing sonar sensors could be used with some modification.
- **Graphical User Interface (GUI):** It is recommended that a graphical user interface (GUI) be made for the implemented programs. This GUI should allow for on-the-fly changes of the control algorithm's custom parameters. This would allow the user to fine tune the system to his/her needs and change them to accommodate the situation. The GUI would also provide information about the system in a simple and easy to understand method. If a camera is ever placed on the end-effector, its video could be displayed in the GUI so that the operator can remotely control Jasper.

There are also other additions that need to be made to the overall system. They are listed as follows:

- **Vision System:** Implementation of a vision system would allow the operator to see what is going on when he/she is out of sight of the manipulator. If the

vision system is located on the end-effector it would make control even more intuitive.

- **Expansion to other mobile manipulators:** Future work involves implementing this algorithm onto an omni-directional mobile manipulator. The omni-directional manipulator will consist of a manipulator mounted on the Omnibot [49]. As opposed to the PowerBot AGV, the Omnibot is holonomic, meaning that it has no constraints on its motion. Applying the current control algorithm should therefore be simpler, due to the lack of nonholonomic constraints. This omni-directional mobile manipulator will have a total of 9-DOF and would therefore be a closer step to full implementation on an ROV (a system with 12-DOF).

References

- [1] John J. Craig, *Introduction to Robotics: Mechanics and Control*, Pearson Prentice Hall, Upper Saddle River, NJ, 3 edition, 2005.
- [2] O. D. I. Nwokah and Y. Hurmuzlu, *The Mechanical Systems Design Handbook: Modeling, Measurement, and Control*, CRC Press, 2002.
- [3] S. Chiaverini, G. Oriolo, and I. D. Walker, *Springer Handbook of Robotics*, chapter 11, pp. 245–268.
- [4] Y. Yamamoto, *Control and Coordination of Locomotion and Manipulation of a Wheeled Mobile Manipulator*, PhD Thesis, University of Pennsylvania, Philadelphia, PA, August 1994.
- [5] tradeindia.com, “Qingdao Ping’an Forging Machinery Co., Ltd.,” On the WWW, May 2009, URL <http://www.tradeindia.com/selloffer/1529126/Mobile-Manipulator-For-Forging-Charging-Discharging-.html>.
- [6] Inc. Flash Tech, “Niosh mining equipment,” On the WWW, May 2009, URL <http://www.cdc.gov/niosh/mining/products/product147.htm>.
- [7] National Institute for Occupational Safety and Health (NIOSH) Mining Division, “Mobile manipulator,” On the WWW, May 2009, <http://www.flashtech-inc.com/Systems/Mobile/Mobile.htm>.

- [8] Kyushu Electric Power Co. Inc., “Mobile manipulator for construction work,” On the WWW, May 2009, http://www.kyuden.co.jp/en_tech_distribution_distribution5.html.
- [9] Neobotix, “Mobile manipulator “mm-500”,” On the WWW, May 2009, <http://www.neobotix.de/en/products/Manipulators.html>.
- [10] American Crane and Equipment Corporation, “teodor,” On the WWW, May 2009, <http://www.americancrane.com/Telerob/tEODor.htm>.
- [11] iRobot, “Government,” On the WWW, May 2009, <http://www.irobot.com/sp.cfm?pageid=109>.
- [12] Shark Marine Technologies Inc., “Remotely operated vehicles,” On the WWW, June 2009, <http://www.sharkmarine.com/rov.htm>.
- [13] Saab Seaeye Limited, “Seaeye panther plus,” On the WWW, June 2009, <http://www.seaeye.com/Pantherplus.html>.
- [14] Science, “Space robot - dextre,” On the WWW, June 2009, <http://www.scienceraay.com/Technology/Space-Robot—Dextre.129460>.
- [15] D. E. Whitney, “Resolved motion rate control of manipulators and human prostheses,” *IEEE Transactions on Man-Machine Systems*, , no. 2, pp. 47–53, 1969.
- [16] A. Liégeois, “Automatic supervisory control of the configuration and behavior of multibody mechanisms,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 7, no. 12, pp. 868–871, 1977.
- [17] C. A. Klein, “Use of redundancy in the design of robotic systems,” in *Proceedings of the 2nd International Symposium on Robotics Research*, Kyoto, Japan, 1984.

- [18] A. Maciejewski and C. A. Klein, “Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments,” *The International Journal of Robotics Research*, vol. 4, pp. 109–117, 1985.
- [19] M. Šoch and R. Lórencz, “Solving inverse kinematics - a new approach to the extended jacobian technique,” *Acta Polytechnica*, vol. 45, no. 2, pp. 165–182, 2005.
- [20] J. Baillieul, “Kinematic programming alternatives for redundant manipulators,” in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, Boston, MA, March 1985, vol. 2, pp. 722–728.
- [21] O. Egeland, “Task-space tracking with redundant manipulators,” *IEEE Journal of Robotics and Automation*, , no. 5, pp. 471–475, 1987.
- [22] L. Sciavicco and B. Siciliano, “Solving the inverse kinematic problem for robotic manipulators,” in *Proceedings of the 6th CISM-IFTOMM Symposium Theory and Practice of Robots and Manipulators*, Krakow, 1987, pp. 107–114.
- [23] K. Tchoń and Ł. Malek, *Advances in Robot Kinematics: Analysis and Design*, chapter 3, pp. 155–164.
- [24] H. Seraji, “An on-line approach to coordinated mobility and manipulation,” in *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, Atlanta, GA, May 1993, vol. 1, pp. 28–35.
- [25] H. Seraji, “A unified approach to motion control of mobile manipulators,” *The International Journal of Robotics Research*, vol. 17, no. 2, pp. 107–118, 1998.
- [26] J. Y. Fourquet and M. Renaud, *Experimental Robotics VI*, vol. 250, chapter 4, pp. 139–149.

- [27] C. C. Wang and V. Kumar, “Velocity control of mobile manipulators,” in *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, Atlanta, GA, May 1993, vol. 2, pp. 713–718.
- [28] J. Tan, N. Xi, and Y. Wang, “Integrated task planning and control for mobile manipulators,” *The International Journal of Robotics Research*, vol. 22, no. 5, pp. 337–354, 2003.
- [29] T. Takubo, H. Arai, and K. Tanie, “Control of mobile manipulator using a virtual impedance wall,” in *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, Washington, DC, 2002, pp. 3571–3576.
- [30] D. Anderson, T. Howard, D. Apfelbaum, H. Herman, and A. Kelly, “Coordinated control and range imaging for mobile manipulation,” in *Proceedings of the 11th International Symposium on Experimental Robotics 2008 (ISER ’08)*, Athens, Greece.
- [31] D. H. Shin, B. S. Hamner, S. Singh, and M. Hwangbo, “Motion planning for a mobile manipulator with imprecise locomotion,” in *Proceedings of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, 2003, pp. 847–853.
- [32] F. G. Pin and J. C. Culioli, “Optimal positioning of combined mobile platform-manipulator systems for material handling tasks,” *Journal of Intelligent and Robotic Systems*, vol. 6, no. 2, pp. 165–182, 1992.
- [33] K. Gurney, *An Introduction to Neural Networks*, CRC Press, 1997.
- [34] C. Gao, M. Zhang, and L. Sun, “Motion planning and coordinated control for mobile manipulators,” in *Proceedings of the 9th International Conference on Control, Automation, Robotics and Vision*, Singapore, 2006.

- [35] S. Lin and A. A. Goldenberg, “Neural-network control of mobile manipulators,” *IEEE Transactions on Neural Networks*, vol. 12, no. 5, pp. 1121–1133, 2001.
- [36] M. W. Chen and A. M. S. Zalzal, *Neural Network Based Motion Control and Applications to Non-holonomic Mobile Manipulators*.
- [37] G. Antonelli and S. Chiaverini, “Task-priority redundancy resolution for underwater vehicle-manipulator systems,” in *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*, Leuven, Belgium, 1998, pp. 768–773.
- [38] G. Antonelli and S. Chiaverini, “Fuzzy redundancy resolution and motion coordination for underwater vehicle-manipulator systems,” *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 1, pp. 109–120, 2003.
- [39] F. Lizarralde and J. T. Wen, “Quaternion-based coordinated control of a subsea mobile manipulator with only position measurements,” in *Proceedings of the 34th Conference on Decision & Control*, New Orleans, LA, December 1995, pp. 3996–4001.
- [40] D. Willis, S. Nokleby, and R. Pop-Iliev, “Mechanical design of the Jasper mobile manipulator,” in *Proceedings of the 2008 ASME International Design Engineering Technical Conferences*, Brooklyn, USA, 2008.
- [41] Thermo Electron Corporation, *CRS F3 Robot User Guide*, 2004.
- [42] ActivMedia Robotics, *PowerBotTM AGV Operations Manual*, 5th edition, 2005.
- [43] Mobile Robots Inc., “Powerbot,” On the WWW, June 2009, <http://www.activrobots.com/press/powerbot.jpg>.
- [44] Mobile Robots Inc., “Research robot bases,” On the WWW, June 2009, <http://www.activrobots.com/ROBOTS/specs.html>.

- [45] P. Sobejko, “Implementation of Redundantly-Sensed Parallel-Manipulator-Based 6-DOF Joysticks,” MSc Thesis, University of Victoria, Victoria, BC, 1999.
- [46] P. Spellucci, *DONLP2-INTV-DYN USERS GUIDE*, Technical University at Darmstadt, Darmstadt, Germany.
- [47] J. Angeles and C. S. López-Cajún, “Kinematic isotropy and the conditioning index of serial robotic manipulators,” *The International Journal of Robotics Research*, vol. 11, no. 6, pp. 560–571, 1992.
- [48] CRS Robotics Coporation, *RAPL-3 Language Reference Guide*, 2000.
- [49] S. Bemis, B. Riess, and S. Nokleby, “Design and control of an Omnibot autonomous vehicle,” in *Proceedings of the 2008 ASME International Design Engineering Technical Conferences*, Brooklyn, USA, 2008.