

A Modular, Behaviour-Based Hierarchical Controller For Mobile Manipulators

Kelvin Gong

A thesis submitted in partial fulfilment
of the requirements for the degree of
Master of Engineering
in
Electrical and Computer Engineering
at the
University of Canterbury,
Christchurch, New Zealand.

28 September 2013

ABSTRACT

A mobile manipulator is a robotic system consisting of a robotic manipulator mounted onto a mobile base. This greatly extends the workspace of the robotic manipulator and allows it to perform more tasks. However, combining both systems increases the complexity of the control task as well as introducing additional controller tasks such as coordination of motion, where executing the task can involve using both the mobile base and manipulator, and cooperation of task, where many tasks can be executed at once.

In this thesis a controller for a mobile manipulator is developed from smaller, simple controller blocks, allowing the controller to be flexible, easy to understand, and straightforward to implement using well-known embedded software implementation approaches. A behaviour-based approach was used to build the individual controllers, and a hierarchical structure was used to organise the individual controllers to provide cooperation between them and coordinated motion.

The task assigned to the controller was to reach a series of waypoints in a large workspace, while satisfying performance metrics of manipulability and tip-over stability. The operation of the controller was tested in simulation using 100 randomly generated scenarios consisting of five randomly generated waypoints in each. Using default thresholds for manipulability and tip-over stability, the controller was successfully able to complete all scenarios. Further simulations were then performed testing the effects of varying the thresholds of the performance metrics to explore the tradeoffs involved in different parameter choices. The controller was successful in a majority of these scenarios, with only a few failing due to extreme threshold choices. The reasons for these failures, and the corresponding lessons for robot designers are discussed.

Finally, to demonstrate the modularity of the controller, an obstacle avoidance controller was added and simulation results showed the controller was capable of avoiding obstacles while still performing the same tasks that were used in previous tests.

Successful simulation results of the controller across a range of performance metrics shows that the combination of a behaviour based and hierarchical approach to mobile manipulator control is not only capable of producing a functional controller, but also one that is more modular and easier to understand than the monolithic controllers developed by other researchers.

ACKNOWLEDGEMENTS

I wish to express my gratitude to my supervisors Dr. Allan McInnes and Dr. Paul Gaynor for their guidance and help. I am grateful for their professional and technical knowledge as well as their editorial support which has enabled this thesis to be successfully completed. I wish to say special thanks to Dr. Allan McInnes who co-authored a conference paper and book chapter with me, and provided me with the opportunity to attend and present the paper at ICARA 2011.

I would also like to thank the University of Canterbury for the funding towards this thesis and all the helpful staff here in the Department of Electrical and Computer Engineering.

Finally, I would like to thank my family and friends, as well as fellow postgraduate colleagues for all their help and support. Special mention goes to everyone in my office, whom I shared a lot of fun times with both in and out of the workplace.

CONTENTS

CHAPTER 1	INTRODUCTION	1
CHAPTER 2	BACKGROUND	5
2.1	Behaviour-Based Robotics	6
2.1.1	Subsumption Architecture	7
2.1.2	Motor Schemas	8
2.2	Hierarchical Control	9
2.3	Mobile Manipulators	11
2.3.1	Mobile Robots	12
2.3.2	Robotic Manipulators	13
2.3.3	Mobile Manipulators	14
2.3.3.1	Mobile Manipulator: Manipulability and Stability	15
2.4	Summary	17
CHAPTER 3	MOBILE MANIPULATOR MODEL	19
3.1	Mobile Base Model	20
3.1.1	Coordinate System	21
3.1.2	Dynamics	22
3.1.3	Model Parameters	23
3.2	Manipulator Model	24
3.3	Model Implementation	26
3.4	Model Integration	27
3.5	Summary	29
CHAPTER 4	MOBILE MANIPULATOR CONTROLLER	31
4.1	Performance Metrics	32
4.1.1	Manipulability	33
4.1.2	Stability	34
4.2	High-Level Controllers	37
4.2.1	Objective Controller	38
4.2.2	Manipulability Controller	39
4.2.3	Stability Controller	40
4.3	Controller State Machine	43
4.4	Low-Level Controllers	45

4.4.1	Base Movement Controller	45
4.4.2	Manipulator Movement Controller	47
4.5	Integrated System	49
4.6	Extending the Controller	51
4.6.1	Modular Controller Requirements	51
4.6.2	Obstacle Avoidance Controller	53
4.6.3	Controller Integration	56
4.7	Summary	57
CHAPTER 5	SIMULATION RESULTS	59
5.1	Simulation Setup	59
5.2	Baseline Simulation	62
5.3	Stability Threshold Effects	68
5.3.1	Low and High Stability Thresholds	69
5.3.2	No Stability Hysteresis	77
5.4	Manipulability Threshold Effects	80
5.4.1	Low and High Manipulability Thresholds	80
5.4.2	No Manipulability Hysteresis	88
5.5	Extended Controller: Obstacle Avoidance	90
5.6	Summary	93
CHAPTER 6	CONCLUSION	97
6.1	Thesis Summary	97
6.2	Limitations and Future Work	98
REFERENCES		101

Chapter 1

INTRODUCTION

Mobile manipulators are becoming an increasingly popular area of robotics research, and are also of great interest for both commercial and scientific applications. A mobile manipulator consists of a robotic manipulator mounted onto a mobile platform of some kind. The mobile platform dramatically extends the workspace of the manipulator, as it is no longer fixed in one position.

Currently, mobile manipulators are typically used in an exploratory role requiring some manipulation capabilities. Examples of this include search and rescue, bomb disposal and space exploration as shown in figure 1.1. A common theme shared by these examples is the use of mobile manipulators in environments where it is either impossible or unsafe for humans to perform the required tasks.

An emerging market for mobile manipulators is in the service sector, to be used as helper robots to assist the elderly or disabled people around the home or perform simple tasks in the workplace. Industrial applications such as manufacturing can also benefit from the use of mobile manipulators, which provides a greater amount of flexibility in the production chain [Bøgh et al 2012]. This flexibility is required more and more often with the increasing and various amount of products some companies now produce and also with the shift in paradigm from mass production to a more customised production model [Kotha 1995].

Research in the areas of manipulators and mobile platforms is well established. However mobile manipulator research is relatively new, with one of the first published studies on the integration of the two platforms done in 1989 [Carriker et al 1989]. The difficulty that mobile manipulator research seeks to address is the coordinated control of the base and manipulator together. The key control challenges that a mobile manipulator controller faces are:

- Achieve the main task,
- Maintain stability,
- Maintain manipulability,

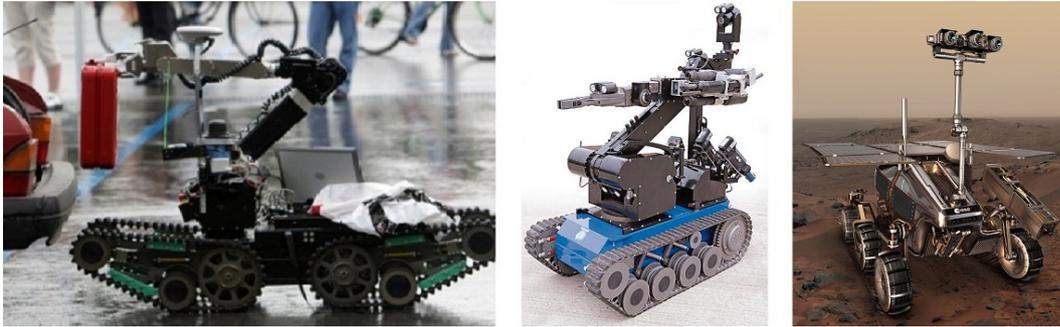


Figure 1.1: Examples of mobile manipulator robots. A search and rescue robot from RoboCup 2009 [RoboCup 2009 2009], tEODor, a bomb disposal robot from TELEROB [Telerob 2010] and the Mars rover [NASA 2010]

- Other tasks, such as obstacle avoidance.

Not only does a mobile manipulator controller have to perform these tasks, the tasks must all be done simultaneously in a coordinated manner.

This thesis proposes a modular, hierarchical mobile manipulator controller built from simple and commonly used controllers to aid in the coordinated motion of the mobile manipulator. The controller design is inspired by behaviour-based robotics systems, which have the benefit of being modular with specific controllers dealing with specific goals independently. This also means the controller can be expanded to increase its capabilities without redesigning the original controller modules. Behaviour-based robotic systems are purely reactive, and the same is true of the proposed controller. The performance of the controller is evaluated using the stability and manipulability performance metrics. A summary of the contents of each chapter is provided below.

- Chapter 2 - Provides background information in the field of mobile manipulators and the previous work that has been performed in this area.
- Chapter 3 - Describes the kinematics, dynamics, and assumptions for the model used to simulate the controller. The overall model is made up of two parts; the mobile base and the manipulator which is then combined.
- Chapter 4 - Introduces the performance metrics of stability and manipulability used to evaluate and provide feedback for the controller. The proposed hierarchical and modular controller design is described along with specific sub controllers tasked with maintaining manipulability and stability, reaching target positions, and obstacle avoidance.
- Chapter 5 - Simulation results are presented to show the effectiveness and operation of the controller while performing tasks comprising of reaching a series of waypoints in different scenarios.

- Chapter 6 - Conclusion of the work that has been performed.

The following publications have been created as a result of this research:

- A paper titled 'A Hierarchical Control Scheme for Coordinated Motion of Mobile Manipulators' presented at ICARA 2011 [Gong and McInnes 2011].
- A book chapter titled 'A Modular Hierarchical Control Scheme for Mobile Manipulation' to be published in 'Recent Advances in Robotics and Automation' in 2013 [Gong and McInnes 2013].

Chapter 2

BACKGROUND

Research in robotics dates back to at least the 15th century, with Leonardo da Vinci producing designs for the first humanoid robot [Rosheim 2006]. However it was only in the 1940s and 1950s that robotics research started to take off, enabled by the computer revolution. The first commercial robot was developed by Unimation, founded by George Devol and Joseph Engelberger [International Federation of Robotics 2012]. The robot was a programmable arm that was eventually installed in a General Motors assembly line in 1961. Since then, the robotics industry has grown exponentially and is an important industry driving economic growth. Now robots can even be found in households, with one of the most popular domestic robot being the Roomba [iRobot 2012] automated vacuuming robot.

The main application of robotics was initially dominated by the machine tool industry. Therefore, early robotics design philosophy was to design linkages to be as stiff as possible with each joint controlled independently as a single-input/single-output linear system. The main barriers to robotic control progress in the early stages were the high computation costs, lack of adequate sensors, and minimal understanding of fundamental robot dynamics [Spong and Fujita 2011]. As these barriers eventually subsided due to technological advances and additional research, more sophisticated control methods integrating force and vision systems became possible, which in turn enabled more complex applications.

Robotics is now a broad and diverse discipline, too large to describe fully here. Instead, this chapter focuses on providing background information on robotic control strategies and frameworks, mobile manipulators, and specific mobile manipulator research relevant to this thesis.

2.1 BEHAVIOUR-BASED ROBOTICS

Two contrasting approaches exist to robotics control:

1. Top-down,
2. Bottom-up.

The top-down approach is associated with heavy planning and is not very flexible, starting with high level goals and decomposing them into sub tasks [Crespi et al 2005]. A classic example of the top down approach is the sense-plant-act methodology, which starts with gathering sensor data to create a world model, from which the next action can be planned and executed. The bottom-up approach starts with a collection of independent sense-act couplings executing concurrently, monitoring sensor values and triggering actions. An example of a bottom-up approach are reactive architectures where sensors directly determine actions. Behaviour-based robotics is a reactive architecture and this approach will be focused on since it is of relevance to this thesis.

Behaviour-based robotics is an approach to robot control that is based on the idea of robotic behaviours that generate motor responses from external stimuli [Arkin 1998]. Typically a behaviour consists of a simple sensorimotor pair providing the most basic of functions. An example of a sensorimotor pair is a proximity sensor, that when triggered, causes the robot to back off from an object. Behaviours serve as the building blocks for more complex robotic actions.

Behaviour-based systems are inherently modular, as an individual module equates to a certain behaviour. This allows simple behaviours to be reused as part of other complex robotic actions to achieve more abstract goals. As behaviours are a reaction to some stimuli, a behaviour-based system is a purely reactive system and avoids the use of abstract representational knowledge. This characteristic is particularly important in dynamic environments where a world model would be impossible to produce. Building world models is also a time-consuming task, and relies heavily on the accuracy of the robot's sensors to create an accurate model. The behaviour-based approach contrasts with the traditional sense-plan-act paradigm used in early autonomous robots, which relies on the planning stage to achieve its goals and avoid problems.

There are no formal rules for defining the right behavioural building-blocks for a robotic system, but a common approach is ethologically guided: animal behaviours are studied and decomposed to find specific behaviours that would be useful in a robotic system. In all behaviour-based systems, there will be a time when conflicting results are produced from multiple behaviours. Dealing with these conflicts, and thus the coordination of many behaviours, is what creates the overall robotic system and can be broken down to two approaches [Arkin 1998]:

1. Competitive: a winner takes all approach. After some weighting of importance, a single behaviour is selected and executed.
2. Cooperative: combines the output of all the behaviours. Weightings are used to give more strength to any particular behaviour based on desired performance characteristics.

Extensive work on behaviour-based robotics has been done since it was first introduced. Arkin [Arkin 1998] provides a thorough survey of research done in the field up to 1998. Current research involving behaviour-based robotics primarily concerns the application of behaviour-based approaches to new and complex applications such as robot learning [Elgawi 2009] [Ray et al 2011].

An early architecture for behaviour-based robotics design is the subsumption architecture developed by Rodney Allen Brooks [Brooks 1986]. This architecture popularised and demonstrated the strengths of behaviour-based robotics. The subsumption architecture is further explored, as well as another behaviour-based architecture, motor schemas, in the next sections.

2.1.1 Subsumption Architecture

Behaviour-based robotics was popularised in the 1980s by Professor Rodney Allen Brooks at the Massachusetts Institute of Technology (MIT). He, with the help of students and colleagues, built a range of both wheeled and legged robots using the subsumption architecture, which he also developed. The subsumption architecture decomposes complex behaviours into simple modules and organises them into layers. Each layer is then tasked with specific objectives, with objectives in the higher levels becoming more abstract. Higher level behaviours take into account the decisions of the lower level behaviours. Response encoding is predominantly discrete; for example the behaviour response to an obstacle on the right of the robot is to turn to the left. Coordination in subsumption uses the competitive approach and has two primary mechanisms:

1. Inhibition: used to prevent a signal from being transmitted to the actuators.
2. Suppression: prevents the current signal from being transmitted and replaces it with the suppressing message.

A stimulus-response (SR) diagram for a subsumption-based foraging robot can be seen in figure 2.1. The task for the foraging robot in this example is to:

- Move away from a home base area and look for objects of interest,
- Avoid any collisions,



Figure 2.1: SR diagram for a subsumption based foraging robot.

- Detect and pick up an object,
- Return to home base.

Priority based arbitration is used as the coordination mechanism with higher level behaviours being able to suppress outputs of the lower level behaviours. A typical rule set to accomplish this task can be to:

1. Start in the move around behaviour to look for objects.
2. The avoid collisions behaviour suppresses the move around behaviour to avoid a collision when required.
3. The acquire behaviour suppresses the move around behaviour once an object is detected.
4. The homing behaviour suppresses the move around and acquire behaviour to return the object back to base.

The behaviour-based approach produces good anthropomorphic qualities in the robots while keeping development costs relatively low as it allows individual behaviours to be tested and incrementally added to form a more complex mobile robot system.

2.1.2 Motor Schemas

Another behaviour-based architecture, motor schemas, was developed by Ronald Arkin shortly after the subsumption architecture [Arkin 1998]. It is more motivated by the biological sciences, with a schema containing the information on how to react and the way that reaction can be realised. Response encoding is continuous, with behavioural outputs presented in a single uniform format of vectors generated from a potential field analog [Khatib 1985]. Potential fields represent a continuous navigational space through the world, based on an arbitrary potential function, with objectives treated as attractors and obstacles repulsors. Coordination is performed using the cooperative

approach by means of vector addition. This is possible as the output of each behaviour is in the same format. Pure arbitration is not used as each behaviour contributes to the overall robot response, with the relative strengths of each behaviour determining what the overall robot response will be. Revisiting the foraging robot example presented earlier, a SR diagram using the motor schema architecture can be seen in figure 2.2.

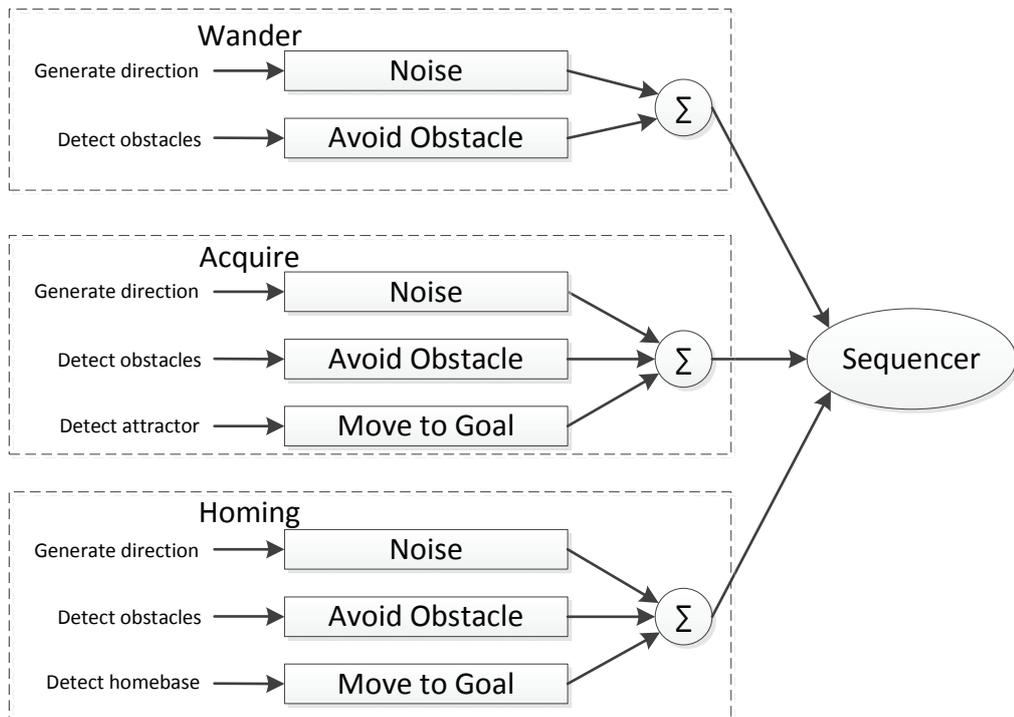


Figure 2.2: SR diagram for a motor schema based foraging robot.

Motor schemas have been applied to a mobile manipulator system in the application of drum sampling [MacKenzie and Arkin 1996] to show the feasibility of this type of control strategy in real-world problems.

2.2 HIERARCHICAL CONTROL

A hierarchical controller is often used to organise a system with complex behaviours. Commands flow down from the top node to the subordinate nodes, with sensors and results flowing back up the chain [Arkin 1998]. Typically, higher levels perform the planning and execution of tasks while the lower levels execute the tasks themselves. A traditional hierarchical structure commonly used throughout 1980s in the field of robotics was the sense-plan-act structure [Jitendra and Raol 2012]. This required a world model to be built from the sensory data, from which the next move can be planned and made. The major limitation of this method is the difficulty in planning and creating the world model accurately.

An example of a hierarchical controller using an architecture that is more reactive and involves planning is the three-layer architecture [Gat 1998]. The three-layer architecture was proposed as an alternative mobile robot architecture to the traditional sense-plan-act approach. The three major components are a controller, sequencer, and a planner. The layout and command flow for a basic three-layer architecture is shown in figure 2.3.

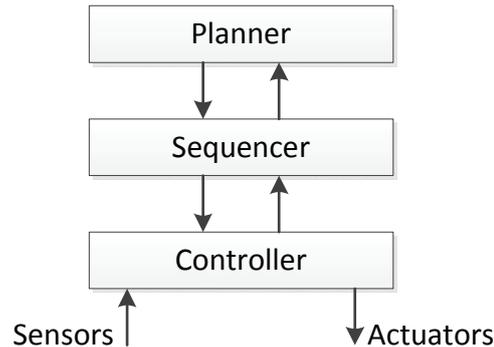


Figure 2.3: Structure of the three-layer architecture.

The three-layer controller consists of a collection of feedback control loops for the various actuation requirements of the robotic system. Important architectural constraints on the algorithms that make up the transfer functions for the feedback control loop include:

- Constant-bounded time and space complexity. The constant should be low enough to provide adequate bandwidth for stable control. That is, the controller needs to react fast enough to dynamic environments.
- Controllers should be designed to detect when they fail rather than to avoid failure. Controller failures can then be informed to the sequencer or planner.
- Avoid use of internal states if possible. Any internal states that are used should not introduce discontinuities as it is the role of the sequencer to manage transitions.

The job of the *sequencer* is to select which control loops the controller needs to execute at a given time and to provide all of the controller parameters. The most difficult task is how to decide which feedback loop to execute. One approach is to identify all the possible states a robot can be in and compute beforehand which specific controller to use for that state. Drawbacks for this method are that it is not always possible for a robot to know its current state, and that the approach disregards the robot's execution history, which can provide additional information. In the basic case, the sequencer receives one method or plan from the planner which is then passed down to the controllers for execution. An extension to this basic approach is reactive action

packages (RAPs) [Firby 1987], which group together all the known methods to execute a task in different situations and pass those to the sequencer. Instead of a perfect plan, the sequencer receives a package of all the plans that could work for the current task in different situations. Along with the collection of plans is a success criteria test to check against. Special programming languages have been developed for the implementation of RAPs.

The *planner* is the top layer of the architecture and consists of the most time consuming tasks, such as performing task planning, vision algorithms, or mapping. The planner is run separately, such that several control loops could have been executed before a result is achieved from the planner. Because of these features, there are no architectural constraints on the planner, which is invariably implemented using standard programming languages.

2.3 MOBILE MANIPULATORS

A mobile manipulator is a robotic manipulator, typically a robotic arm, mounted on to a moving platform. Mobile manipulators are a natural progression in the field of robotics, combining the existing fields of mobile robots and robotic manipulators. Robot manipulators were predominantly used in an industrial environment and perceived as such, but the advent of mobile manipulators has shifted this viewpoint towards a intelligent robotic system potentially capable of being an every day “life partner” [Christensen 2009].

As mentioned in Chapter 1, manufacturing can also benefit from the use of mobile manipulators. This has been experimentally tested in automotive manufacturing [Hamner et al 2010]. Hamner combined both visual and force servoing and coordinated motion to successfully perform a peg-in-hole type assembly task. The “Little helper” [Hvilshøj et al 2009] is another industrial mobile manipulator platform developed at Aalborg University, Denmark. A prototype has been built and demonstrated in an imitation production setting, with future focus on usability, robustness and safety. The University of Massachusetts Amherst is developing an autonomous mobile manipulation platform [Katz et al 2006] from off-the-shelf components, with future plans to develop perception, manipulation and tool use of the system.

The two parts—manipulator and mobile base—have been well researched individually in the robotics field. The following sections provide background information on specific areas in these fields relevant to this thesis. Recent research specific to mobile manipulators are also considered.

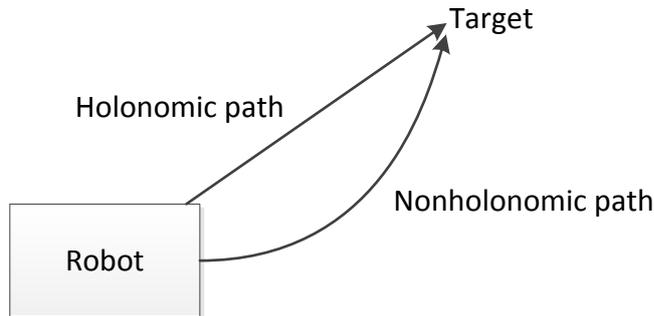


Figure 2.4: Example paths of holonomic and nonholonomic robots.

2.3.1 Mobile Robots

A mobile robot is essentially any robot that moves, but usually describes a robot that can move around in an area but possesses little or no capabilities for manipulating the environment. Mobile bases, a subset of mobile robots, are mobile robots designed to serve as an operating platform for additional hardware. An example of a commercially available mobile base is the Pioneer-3DX [MobileRobots 2012], which provide a large area for the mounting of additional hardware such as actuators and sensors.

A common constraint associated with mobile robots is the nonholonomic nature of the system. That is, the controllable degrees of freedom are less than the actual degrees of freedom. What this means is that not every path is available to the robot. For example, an automobile has three degrees of freedom, consisting of its orientation relative to a fixed heading and its position in two axes. However, the only controllable degrees of freedom are the acceleration and the steering angle. The car's heading must be aligned with the orientation of the car and cannot move in any other direction (assuming no skidding). As a result of the nonholonomic constraint, the car cannot directly move left or right, but has to turn and take a less efficient path, as shown in figure 2.4. Typical vehicles with nonholonomic constraints are those with normal wheels or tracks on either side and using skid or explicit steering [Shamah 1999]. Holonomic systems are possible through the use of omnidirectional wheels such as the Mecanum wheel [Adascalitei and Doroftei 2011] but such systems are usually very costly.

Control of nonholonomic mobile robot systems is a well-researched field. Kolmanovsky et al. [Kolmanovsky and McClamroch 1995] provide a summary of the developments and control strategies in this area. The main control strategy discussed by Kolmanovsky is based on motion planning, with popular topics being optimal motion planning and motion planning involving obstacle avoidance. Motion planning is concerned with obtaining an open loop solution that controls a nonholonomic mobile robot from an initial state to a final state. This is not an easy task compared to controlling holonomic systems, where a set of independent generalised coordinates can be found to move from the initial to final state because motion in any arbitrary direction

is possible. In contrast, motion planning requires obeying the instantaneous motion constraints of the robot, that is, the planned path cannot contain sideways sliding. Nonholonomic constraints are still the main issue in mobile robot design today. Because it is a mechanical constraint it will always be present, and so will always need to be considered.

2.3.2 Robotic Manipulators

Robotic manipulators come in all sorts of different configurations depending on the application. The most easily recognisable type is the articulated robot, which is a robot with rotary joints. Articulated robots usually have six degrees of freedom and at least three rotary joints, as shown in figure 2.5. They are used for a range of applications, from factory assembly lines to welding and in space exploration. An extensive collection of control strategies for manipulators can be found in the book *Robot Manipulator Control Theory and Practice* [Lewis et al 2004].

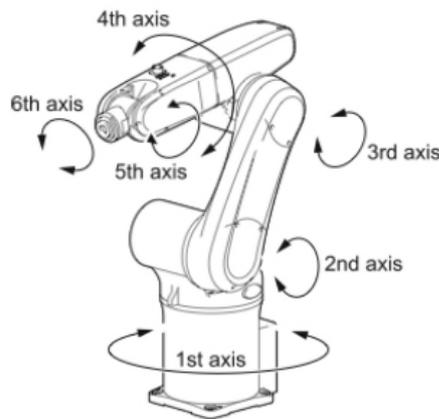


Figure 2.5: Six axis articulated robotic arm [Denso 2009].

The most common class of manipulator controllers are computed-torque controllers, from which more advanced control strategies such as adaptive and learning control have been derived [Lewis et al 2004]. Computed torque is a “classical” dynamic control technique whereby the demand torque for each joint of the robot is calculated based on current joint angles, joint angle rates, and desired joint angle acceleration. Fundamental to the computed torque control strategy is the assumption that the model of the robot and workspace is accurate. This represents a problem as this assumption is sometimes not valid. Some parameters, even in well structured industrialised locations, are difficult to obtain accurately, or difficult to obtain accurately in a timely manner. An example of this would be the dynamic properties of something that was just grasped by the manipulator. Parametric uncertainties greatly affect the accuracy of the controller in high speed applications, and adaptive control strategies have been proposed to improve

this. The adaption mechanism continuously tracks the error and reduces it with time, giving consistent performance in varying load conditions.

2.3.3 Mobile Manipulators

Compared to mobile bases and robotic manipulators, mobile manipulators are a relatively new branch of robotics research. The difficulty in mobile manipulator research arises from the complexity of the integrated system. The dynamics of the manipulator and mobile platform not only differ, but also interact. The addition of a mobile base can introduce nonholonomic constraints, as well as kinematic redundancy a controller needs to resolve. Challenging tasks for mobile manipulator controllers to deal with include coordination of motion and cooperation between tasks [Wasik 2004].

Mobile manipulator research initially focused on issues such as dynamic interaction of the two parts [Yamamoto and Yun 1994] and path planning [Carriker et al 1991] [Dubowsky and Vance 1989]. A recent trend in mobile manipulator research has been the application of neural networks to identify the unknown or changing robot dynamics. Sheng Lin, in his thesis [Lin 2001], developed a neural-network based hierarchical controller to take into account kinematic constraints and adapt to uncertain disturbances. Chen [Chen et al 2006] has also used a neural network based approach to identify unknown system parameters and perform learning, and Singh [Singh and Sukavanam 2011] used neural networks to develop a motion/force control scheme for mobile manipulators with model uncertainties and external disturbances. The use of neural networks or other types of adaptive methods is popular in mobile manipulator research due to the changing and uncertain dynamics and disturbances.

Although mobile manipulators introduces new challenges, the extra redundancy in the mobile manipulator system also enables any controller to control for and improve new performance metrics, namely manipulability and tip-over stability. These performance metrics are either not applicable or cannot be improved without completely forgoing task execution when applied separately to mobile bases or robotic manipulators. Not only can the new performance metrics be used to measure and evaluate the performance of the mobile manipulator, the key tasks of the mobile manipulator will include controlling to improve the performance metrics [Wasik 2004]. More detail of these performance metrics and the implementation of them in this research is described in Chapter 4. Different researchers have tackled different aspects of the mobile manipulation problem. The next section reviews the research that has focused on addressing manipulability and stability and discusses how the work done in this research aims to improve on the prior work.

2.3.3.1 Mobile Manipulator: Manipulability and Stability

Yamamoto [Yamamoto and Yun 1995] developed a controller to coordinate locomotion and manipulation of a mobile manipulator using manipulability as a performance metric to evaluate the performance of the controller. The goals of their research are very similar to the ones in this research, however, this research expand further by including more performance metrics. Their model of a mobile manipulator consisted of a simple two-wheeled mobile platform using skid steering and a two-link planar manipulator. The feedback control was implemented through input-output linearisation with the output equation due to the state equation not input-state linearisable. The goal of the feedback controller was to track a trajectory while maintaining a preferred configuration. Simulation results showed their controller was able to track two trajectories, a straight line perpendicular to the forward direction, and one at 45° slant while maintaining manipulability to be at 1 (maximum) for most of the simulation. Complex trajectories were not simulated, and once the mobile base was moving in the right direction the manipulator was nearly static. Even though the preferred configuration was always maintained in their simulations resulting in maximum manipulability, there exist times where it is not necessary to have perfect manipulability but some lower threshold such that task execution can be done without having to constantly adjust to maintain manipulability. Their controller design was also monolithic, which makes adding or changing performance capabilities difficult. The proposed controller in this research aims to improve on these areas, particularly the monolithic controller design where the proposed controller structure is very modular, which allow extra capabilities to be added to the controller.

Nagatani et al [Nagatani et al 2002] addressed manipulability through a combination of path and motion planning for the mobile base and manipulator respectively. Nagatani et al. calculated the distribution of manipulability for any particular pose of the mobile manipulator. A manipulability area is then defined as a plane sliced from the distribution of manipulability, with the height of the slice in the distribution representing the manipulability threshold. The manipulability area thus show areas where manipulability is above or below the threshold. To complete their proposed task of drawing line segments on a wall, the required poses are all calculated beforehand along with their manipulability areas. A path can then be planned that only travels through regions in the manipulability area that is above the manipulability threshold. Simulation results indicated the motion planning algorithm was able to maintain the manipulability and perform the task. Unfortunately in experimental results, positional errors caused the mobile base to stray from the planned path. The motion planning approach described here contrasts with the controller design in this research which is reactive based.

System stability is another major concern for mobile manipulators, as there are

chances of tipping over during task execution, especially when handling large loads or moving over rough terrain. Methods for addressing stability generally fall into two groups:

1. A reactive approach, using system reconfigurability [Iagnemma et al 2000], whereby the position of the centre-of-mass is altered by moving the manipulator, mobile base, or both. Iagnemma also defines two types of reconfigurability, external and internal. Internal reconfigurability is defined by the contact points of the mobile base remaining fixed relative to the ground during reconfiguration; for example, having some form of suspension system that lets the mobile base tilt. In external reconfigurability the contact points are allowed to move during the reconfiguration process but this also mean the terrain profile will also influence the reconfiguration. For a simple and common mobile manipulator setup that is being considered in this research (without tilting suspensions), external reconfigurability have to be used.
2. Path or trajectory planning [Huang et al 1998] [Furuno et al 2003] to avoid running into stability problems in the first place. A limitation of the path planning approach, especially for mobile manipulators, is the dynamic work space. This can result in requiring to perform lots of replanning which can be computationally expensive and time consuming.

As mentioned previously, the main concern for system stability of a mobile manipulator is in terms of tip-over stability. Stability measurement is thus primarily based on the moments of the system around possible tip-over axes. There are static moments, dynamic moments due to accelerations, and externally induced moments which can all effect the stability measurement. Zero moment point(ZMP) has been used as a stability criterion primarily for biped walking robots [Takanishi et al 1990] [Erbatur et al 2002] and has been applied to mobile manipulators [Huang et al 2000]. The ZMP is defined as a point on the ground where the sum of all moments about that point is zero. If that point is then inside the contact area between the mobile base and the ground, then the mobile manipulator is considered stable. The Force-Angle stability measure [Papadopoulos and Rey 1996] is another method that calculates the net moments about all possible tip-over axes. The axis that is closest to tip-over is then used as the stability measure.

Although most mobile manipulator control research has focused on monolithic controllers, some research concerning a combination of behaviour-based and hierarchical controller structures, has been done [Ogren et al 2000] [Arkin and Mackenzie 1994]. However, neither specifically aims to address manipulability and tip-over stability. Arkin and Mackenzie describe how they integrate world knowledge with reactive control through a hybrid deliberative/reactive system. Ogren applied a strictly

reactive based control for dynamic trajectory tracking while avoiding obstacles with the mobile base.

2.4 SUMMARY

In terms of manipulability and stability, research has been performed primarily focusing on only one aspect at a time. This research proposes a behaviour-based hierarchical controller to address both manipulability and stability. The modularity aspect due to the behaviour-based approach is a departure from the classical control strategies which tends to relate the input and output in a tightly coupled fashion which makes extending the controller difficult.

Chapter 3

MOBILE MANIPULATOR MODEL

Simulation provides a low-cost and flexible way to prototype robot control algorithms like the ones this research seek to develop. In order to test the control algorithms, representative models of the robot are required, and, in this case, a mobile manipulator model. There are no simulation packages specifically targeted at mobile manipulator simulation. Some generic robotics simulation packages can be extended to work with mobile manipulators, such as RobWork [Ellekilde and Jorgensen 2010], an open source toolbox, or the commercially available CAD design and simulation solutions from Dassault Systmes [Dassault Systèmes 2012]. The options were to develop a simulation from scratch or attempt to modify existing simulation packages to include a mobile base. Without knowing what kind of limitations might be encountered when modifying existing packages, the decision was made to create a model from scratch. Creating a model from scratch provides greater flexibility and customisation options in terms of the features and capabilities that are required, and can be tailored for use in this research.

MATLAB Simulink was selected as the modelling and simulation platform due to its all-round feature set and many additional libraries and toolboxes for specific applications. One of these toolboxes is the Robotics Toolbox [Corke 2008] for manipulators developed by Peter Corke. The toolbox provides MATLAB objects defining the properties of links and configuration of a manipulator. Useful mathematical functions related to manipulators, such as homogeneous transforms and rotation matrices, are also provided. Another benefit of using MATLAB to develop the model is that controllers can also be developed in MATLAB as well. This will help the integration of the two parts when performing combined simulations of the robot and controller.

The simulation model of the mobile manipulator was broken into three parts:

- The mobile base model,
- The manipulator model,
- Integration of mobile base and manipulator models.

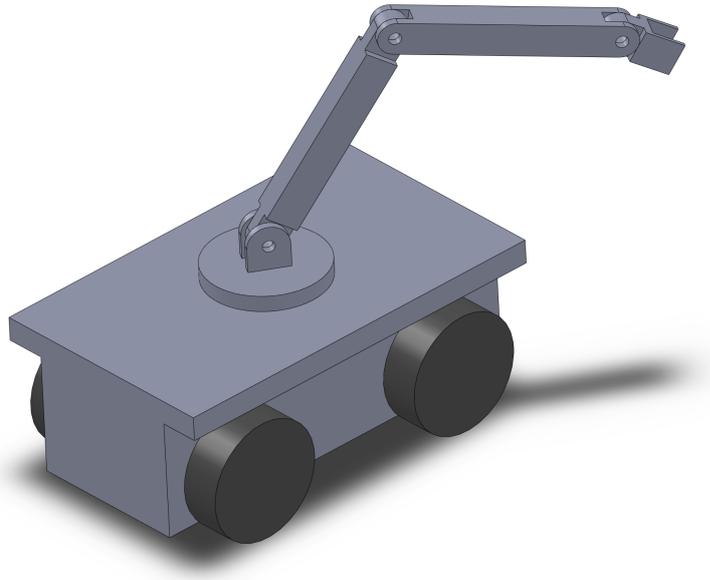


Figure 3.1: CAD model of the mobile manipulator.

A conceptual CAD model of the mobile manipulator is shown in figure 3.1. The following sections describe the individual models, and their integration together.

3.1 MOBILE BASE MODEL

The aim of the mobile base model is to generate behaviour representative of a typical mobile base platform under normal operating conditions. The mobile base model should be able to move and turn when a drive input is applied, and have its behaviour affected by its physical parameters such as mass and inertia in an expected manner. For simplicity, the mobile base is always considered to be on a completely flat surface.

The model for the mobile base is based on a differential-drive configuration (also known as skid steering). This is a common type of configuration for a mobile base and is widely used in research and industry. Differential drive requires the mobile base to be able to independently control the speed of the wheels on either side of the mobile base. Turning is achieved when there is a difference in speed between the wheels as shown in figure 3.2. Differential drive provides better manoeuvrability when compared to the standard methods of steering used in everyday vehicles such as cars and trucks, as it allows the mobile base to be capable of rotating on the spot. Differential-drive configurations are also mechanically less complex than other drive systems as there are no linkages between the wheels and the wheels always point in the same direction. In terms of modelling, this translates to a reduction in the complexity required to produce an accurate model.

Using a differential-drive configuration means nonholonomic constraints are introduced to the system dynamics. From a performance perspective, adding these con-

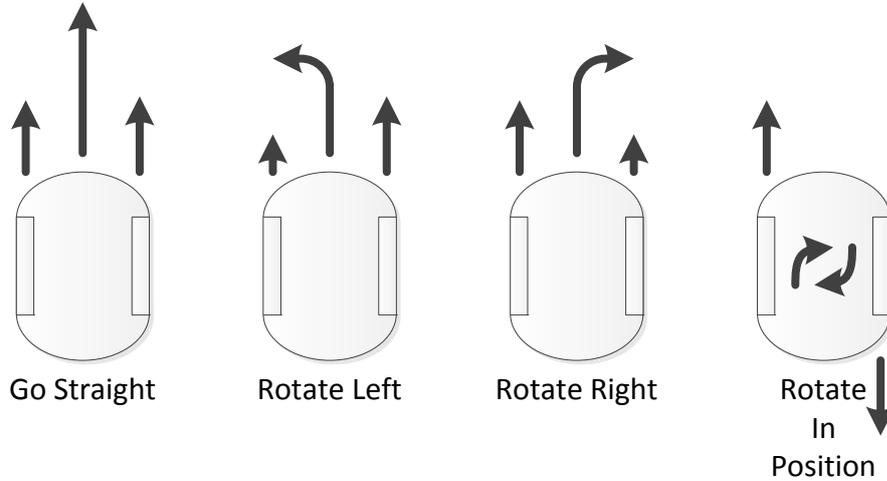


Figure 3.2: Example of a differential-drive system and how turning is achieved.

straints is not desirable as it limits the controllable degrees of freedom of the robot, but from a research perspective adding the constraints is useful, as most mobile base platforms have nonholonomic constraints, therefore this model is more comparable with, and applicable to, real world situations. For the mobile base model used in this research the total degrees of freedom is three: position in the x and y axes, and an orientation relative to a fixed heading. However, the controllable degrees of freedom are limited to only the forward or reverse motion and the rate of steering. No direct motion allowing the mobile base to translate left or right can occur.

Figure 3.3 shows the configuration of the mobile base model, and the locations where force from the wheels are applied to provide the translation and rotational motion. The parameters w , l , h denote the width, length and height of the mobile base, and F_1 and F_2 the forces applied.

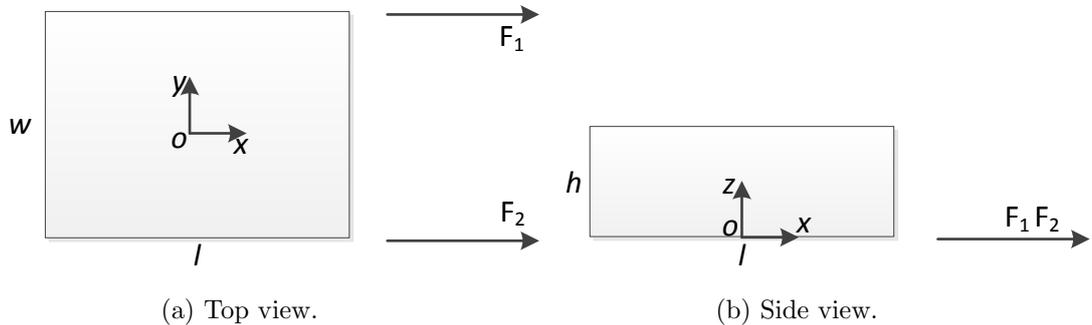


Figure 3.3: Mobile base model and physical parameters.

3.1.1 Coordinate System

The coordinate system used for the mobile base is shown in Figure 3.4, where G denotes the global coordinate frame, L denotes the local coordinate frame centred on the

robot, and R denotes the robot body frame. The global and local coordinate frames are parallel to each other, with the transformation between them representing the xy position of the mobile base. The local and robot body coordinate frames are located in the same position, with the transformation between them representing the heading angle of the mobile base. Together, this allows the location and heading of the mobile base to be described in the global coordinate frame.

Rotations in the anti-clockwise direction about the z -axis are deemed to be positive. The origin of the robot and local coordinate systems are physically located at the centre of the mobile base on the ground plane. The centre of mass of the mobile base is assumed to be at the geometric centre of the robot.

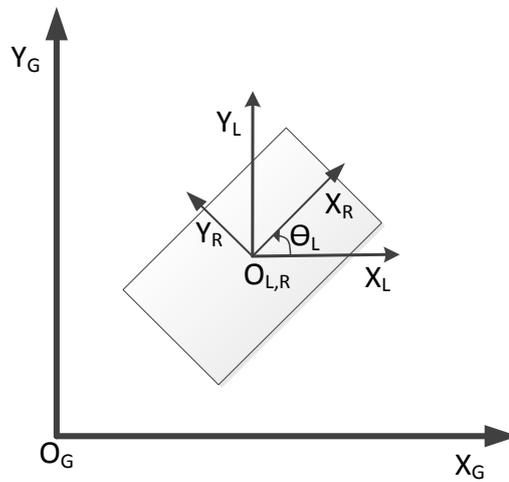


Figure 3.4: Mobile base coordinate system definition.

3.1.2 Dynamics

The equations of motion for the mobile base model are shown in equations (3.1) and (3.2) and are based on Newton's second law with added dampers. They describe the translational and rotational motion of the mobile base. The equations of motion are:

$$m\ddot{x}_R + c_1\dot{x}_R = F, \quad (3.1)$$

$$I\ddot{\theta}_L + c_2\dot{\theta}_L = T, \quad (3.2)$$

where

x_R : x coordinate in the robot frame,

θ_L : robot heading relative to local axis,

F : force generated by the drive system,

T : torque generated by the drive system,

m : mass of the mobile base (includes manipulator mass in the integrated model),

I : inertia of the mobile base about its z -axis (includes manipulator inertia in the integrated model),

c_1 : damping constant for translational motion,

c_2 : damping constant for rotational motion.

The model defined by equations (3.1) and (3.2) is a lumped parameter model, which simplifies the description of the robot. Assumptions associated with this choice of model are that the base is a rigid body, and that all interactions take place via kinematic pairs, springs, and dampers.

The force and torque generated by the drive system are functions of the force generated by each wheel:

$$F = F_1 + F_2, \quad (3.3)$$

$$T = (F_2 - F_1) \frac{w}{2}, \quad (3.4)$$

where

F_1 : Force applied on the left drive system of the mobile base,

F_2 : Force applied on the right drive system of the mobile base,

w : Distance between the left and right drive systems.

An assumption is made that the forces are a direct input to the model from a controller. This eliminates the need for a detailed specification of motors and wheels, and makes it simpler to adjust the model to behave and perform like various commercially available mobile robots where detailed information such specific motor properties are not available. Important dynamic effects such as motor drag are incorporated into the damping constants in the model.

The net force generated from the wheels induces linear acceleration, while an imbalance in forces from each side of the mobile base creates torque and induces angular acceleration. The nonholonomic constraint is implicit in this model, as the equation of motion governing the translational motion of the mobile base only allows motion in the direction defined by the instantaneous heading of the mobile base. Forward and reverse motions are possible and are treated equally in terms of performance.

3.1.3 Model Parameters

The following parameters can be changed to alter the physical characteristics of the mobile base to affect the robot performance:

- Mass,
- Moment of inertia,
- Length, width and height,
- Translational damping constant,
- Rotational damping constant,
- Force limit.

The mass and inertia primarily affect the linear and angular acceleration of the mobile base. Even though the inertia is directly related to the mass, it was not tied to the mass in the model. This provides some flexibility in altering the inertia without changing the mass to see performance differences. The dimensions of the mobile base do not affect the dynamics of the mobile base or its performance in terms of translational motion, but will affect rotational motion as the torque generated is related to wheel separation which, for the model used here, is directly related to the width. The mobile base dimensions are also important when stability is considered in the integrated model.

The purpose of the damping constants is to slow and eventually stop the mobile base if no drive force is applied and also prevent infinite acceleration. Table 3.1 shows the model variables used to create the mobile base used for simulation studies in this research. The physical properties were loosely based on the Pioneer P3-DX [MobileRobots 2012] mobile base platform. The values for the damping constants were experimentally obtained such that maximum translational and angular velocities were comparable to the Pioneer P3-DX given the set force limit. The force limit is set to limit the acceleration of the robot to 1 ms^{-2} .

Table 3.1: Mobile base model variables.

Mass	10 kg
Moment of inertia	0.5 kgm^2
Length	0.45 m
Width	0.35 m
Height	0.2 m
Translational damping constant	15 Nsm^{-1}
Rotational damping constant	$4 \text{ Nms}(\text{rad}^{-1})$
Force limit	20 N total (10 N per wheel)

3.2 MANIPULATOR MODEL

The manipulator model in this research is based on the well-known PUMA560 [Corke and Armstrong H elouvry 1995] manipulator, which has six degrees of freedom (DOF)

and only revolute joints. The PUMA560, or manipulators in this configuration, have been used in many research papers as the manipulator platform. Figure 3.5 shows the PUMA560 manipulator configuration. For this research the size and geometry of the manipulator has been simplified to reduce the complexity of the model. This simplification involves assuming a uniform shape and mass distribution for each of the individual links. Table 3.2 describes the standard Denavit-Hartenberg [Denavit and Hartenberg 1955] and additional physical parameters used to create the model, where

α : link twist angle,

\mathbf{A} : link length,

θ : link rotation angle,

\mathbf{D} : link offset distance,

σ : joint type, used by the Robotics Toolbox to indicate either revolute or prismatic joints,

\mathbf{m} : mass of link.

Table 3.2: Standard Denavit-Hartenberg and physical parameters for the simplified manipulator model.

Link	α (rad)	\mathbf{A} (m)	θ (rad)	\mathbf{D} (m)	σ	\mathbf{m} (kg)
1	$\pi/2$	0	0	0	0	0
2	0	0.35	0	0	0	1
3	0	0.3	0	0	0	1
4	$\pi/2$	0	0	0	0	1
5	$-\pi/2$	0	0	0	0	0
6	0	0	0	0	0	0

The manipulator's link lengths and masses can be varied to simulate different configurations and situations, but the link twist angles are always kept the same as this is the part based on the PUMA560 manipulator. The link offset distance is set to be zero for all links to simplify the model. Mass loads can be added to the end effector of the manipulator by changing the mass of the end effector links. The Robotics Toolbox also allows additional parameters to be included such as the motor inertia, gear ratios and viscous friction, but these were not used as they were not relevant to the controller. The purpose of this research was to design a controller for mobile manipulators and so all that is required from the manipulator model is that it looks like a manipulator with respect to the controller, and responds to control inputs.

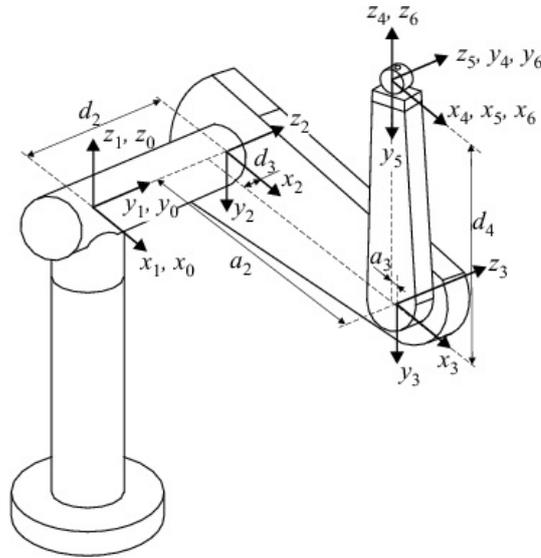


Figure 3.5: PUMA560 manipulator configuration [Kozlowski et al 2003].

3.3 MODEL IMPLEMENTATION

The MATLAB implementation in Simulink of the model is described in this section. A mobile base object was created encapsulating all the physical parameters described previously. The model is contained in a single Simulink block with the inputs and outputs described in table 3.3.

Table 3.3: Inputs and outputs of the mobile base Simulink block.

Inputs	Outputs
Force left	Position
Force right	Velocity
Manipulator inertia	Acceleration
Manipulator mass	Direction
	Angular velocity
	Angular acceleration

The “force left” and “force right” inputs are the driving forces for the mobile base. Imbalance in the driving forces will induce turning. Provisions were made for the manipulator mass and inertia to be included in the mobile base block in anticipation of model integration. If a manipulator is added to the system then its mass and inertia need to be taken into account in addition to the mobile base’s mass and inertia when evaluating the equations of motions.

The position output provided gives the xy position of the mobile base in the global coordinate frame, while the velocity and acceleration outputs provide scalar values relative to the robot coordinate frame. This is useful in determining more clearly; for example, if the mobile base is slowing down or going in reverse. The direction output

provides the heading of the mobile base relative to the local x -axis, which is parallel to the global x -axis. Again the angular velocity and acceleration outputs are in the robot frame, which allows the direction and rate of turning to be quickly determined.

The equations of motions are implemented as a MATLAB function. An excerpt of the MATLAB function is shown in listing 3.1. An input packet is passed to the function, containing of all the necessary arguments required such as the driving forces and current mobile base velocities. The algorithm outputs the accelerations for that time step, which are integrated using Simulink's built-in integrator to calculate the velocity and distance travelled in each time step.

Listing 3.1: MATLAB code for the equations of motion.

```

1 ftotal = F(1)+F(2); %net force
2 T = (F(2)-F(1))*base.w/2; %net torque
3 Vddn = (ftotal-Fv*Vd)/m; %translational acceleration
4 theta_dd = ((T-Fw*theta_d)/I); %angular acceleration

```

The distance d obtained from the integrator is in the robot coordinate frame, in line with the robot x -axis. To translate this back to the global coordinate frame, equations (3.5) to (3.7) are used. The average direction θ_{avg} is used as the direction of travel for a particular time step to improve the accuracy of the estimated global position. The global positions are then stored and used to build up the path of the mobile base for the duration of a simulation.

$$\theta_{avg} = (\theta_{prev} + \theta)/2 \quad (3.5)$$

$$X_G = X_{G_prev} + d \cos(\theta_{avg}) \quad (3.6)$$

$$Y_G = Y_{G_prev} + d \sin(\theta_{avg}) \quad (3.7)$$

The implementation of the manipulator model was straightforward. A robot manipulator object was created using the Robotics Toolbox and configured using the Denavit-Hartenberg parameters.

3.4 MODEL INTEGRATION

To create the mobile manipulator model, the manipulator and base models were integrated. The aim of the integrated model is to provide correct positions of the manipulator and base in the global reference frame, and take into account the combined mass and inertia of the system.

To combine the two models together, the additional mass and rotational inertia in the z -axis has to be taken into account when performing the base movement. The manipulator is placed in the centre of the mobile base on top of the origin of the robot

coordinate frame. The offset height is determined by the height of the mobile base. Although the mass of the manipulator is constant, the inertia is dependent on the pose of the manipulator and thus must be calculated at each step of the simulation. The manipulator mass and inertia are then combined with the mobile base mass and inertia to update the equations of motion. The inertia of the manipulator is calculated using the inertia function provided by the Robotics Toolbox. A uniform mass distribution is assumed for the individual links of the manipulator, which places the centre of mass of each link in the midpoint of each link. The manipulator load is treated as a point mass at the end effector location. Reaction forces from the acceleration of the manipulator are also assumed to be negligible.

To enable the manipulator to move with the mobile base, the joint coordinates need to be translated by the current coordinates of the mobile base as well as rotated by the current heading of the mobile base. Again, the Robotics Toolbox made this task easy by providing both a matrix rotation function (*rotz*) as well as a translation function (*transl*). An excerpt of the MATLAB functions that allows the manipulator to move with the mobile base is shown in listing 3.2. The basic procedure is the creation of a transform matrix that relates the global origin to a particular joint. From these transform matrices the translation function is used to calculate the location of the joint in the global coordinate frame.

Listing 3.2: MATLAB code for the movement of the manipulator with the mobile base.

```

1 arm_base = transl(pos); %base translation
2 arm_base(3,4) = arm_base(3,4) + base.h; %include base height
3 arm_base(1:3,1:3) = rotz(theta*pi/180); %rotation
4 t(:, :, 1) = arm_base; %frame of first joint
5 for i = 1:n %individual transforms to each successive joint
6     t(:, :, i+1) = t(:, :, i)*L{i}(q(i))*arm.tool;
7     joint_coord(i, :) = transl(t(:, :, i+1))'; %locations in xyz
8 end

```

The connected mobile manipulator model is shown in figure 3.6. The outputs of the model are used by a controller, and the inputs are the outputs of that controller, thus closing the loop. A MATLAB 3D plot of the model to be used in visualising the robot's motion is shown in figure 3.7, with the diamond denoting the front of the mobile base.

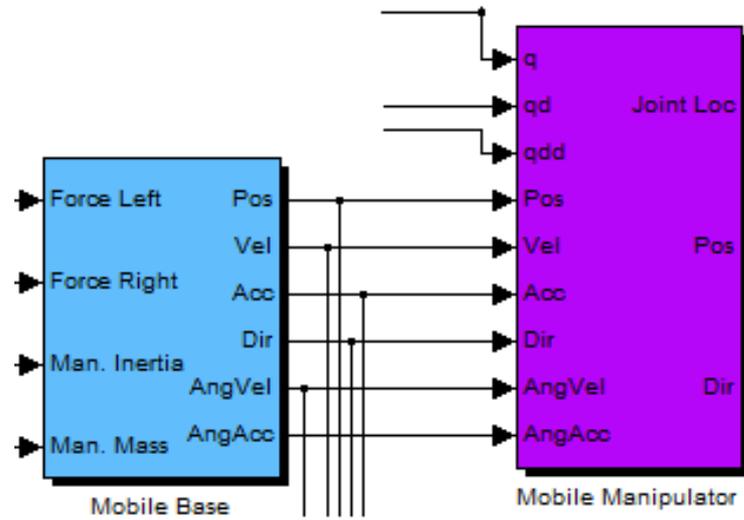


Figure 3.6: Mobile manipulator Simulink blocks.

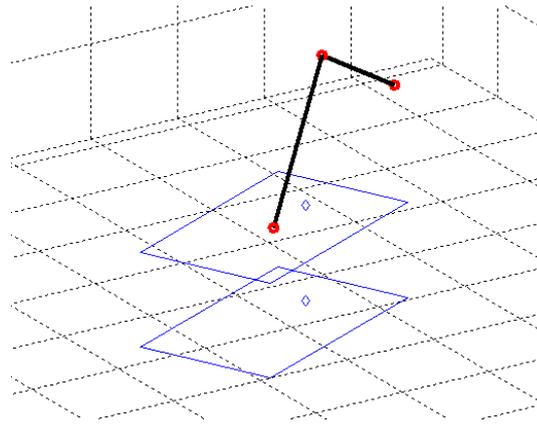


Figure 3.7: Mobile manipulator model in simulation.

3.5 SUMMARY

A nonholonomic mobile base model was created in MATLAB, along with a simplified 6 degree of freedom articulated manipulator created using the Robotics Toolbox. The two models are integrated together to create an overall mobile manipulator simulation model combining the key parameters of system mass, inertia, and correct global positions of the manipulator. This model is used to simulate the proposed controllers from this research, which is discussed in Chapter 4.

Chapter 4

MOBILE MANIPULATOR CONTROLLER

The purpose of the controller is to decide how to execute a series of commands for the mobile manipulator to perform some task, while still maintaining a set of performance metrics above their threshold level. The main goals for the overall controller design were to be modular, flexible, and simple to implement. To meet these goals a behaviour-based controller approach is used to encapsulate individual tasks such as obstacle avoidance or movement to target into independent and modular sub controllers. A hierarchical controller framework is used to organise and categorise the controllers into low and high-level controllers, and to dictate what behaviours are executed and how different behaviours can interact. Defining a standard framework that the behaviours must adhere to allow new behaviours to be added with minimal changes to the other controllers. This not only simplifies the addition of behaviours, but provides flexibility as well. The overall controller design is split into three main sections:

1. A collection of low-level controllers for the control of the direct movement of the mobile base and manipulator to reach a specific target.
2. A collection of high-level controllers that determine where and how to move the mobile manipulator to achieve specific goals.
3. A state machine controller that determines the current goal, and, in turn, selects which high-level controller to execute. These decisions are based on the performance metrics that the controller is required to maintain, as well as the objective of the mobile manipulator.

The performance metrics used to evaluate and provide feedback for the controller are manipulability and stability. These metrics are applied while the mobile manipulator is attempting to reach a series of waypoints as part of its task execution. The following section describes these performance metrics and how they are calculated. Then the individual controllers that address each performance metric are described, and the way that the controllers are integrated and work together is explained. Figure 4.1 shows the structure of the major components of the overall controller and their

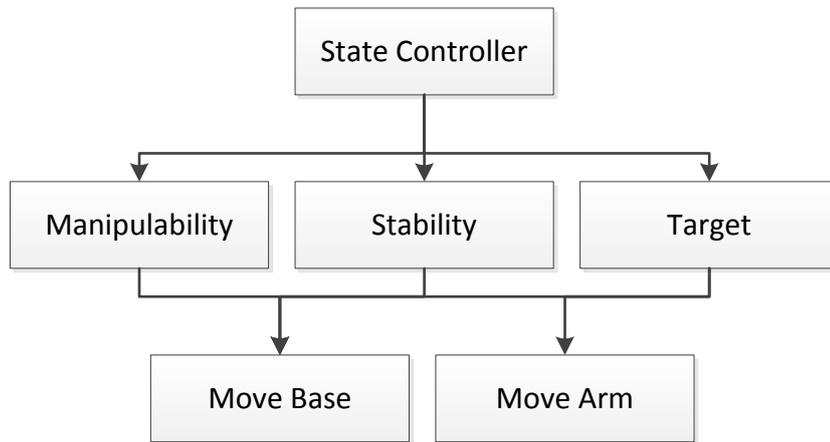


Figure 4.1: Structure of the overall controller.

hierarchy, with the arrows representing moving down the hierarchy. This control structure is loosely based on the three layer architecture [Gat 1998] described previously in Chapter 2. The planning layer of the three layer architecture is not implemented as it was not in the scope of this research.

4.1 PERFORMANCE METRICS

The following two performance metrics are introduced and will be used by the proposed controller:

1. **Manipulability:** gives an indication of how far the manipulator is from singularities, and thus a measure of the manipulator's ability to exert forces in all directions [Yoshikawa 1990]. Manipulability is a well known property in robotic manipulator research. But when used purely in a robotic manipulator system, no mechanisms exist which allow improvements in the manipulability while still maintaining a targeted end effector position, that is, the base position is fixed. A high manipulability improves task execution by ensuring the manipulator can reach all possible directions before a new task is issued.
2. **Tip-over stability:** this is always an issue with moving robots, but the problems are amplified in a mobile manipulator system. The movements of the manipulator constantly change the moments of inertia of the overall system, whereas a typical mobile robot system has a static moment of inertia. In regards to robotic manipulators, tip-over stability is generally not considered, as the manipulators are attached to an immovable platform.

The role of the stability and manipulability performance metrics is to evaluate the current configuration of the mobile manipulator during its operation and to provide feedback for the high-level controllers. As with most systems with a high degree of

freedom, there exists certain preferred configurations for task execution which the performance metrics will be used to define.

4.1.1 Manipulability

Manipulability is a measure of a manipulator's ability to exert forces in all directions. The standard manipulability metric [Yoshikawa 1990] is used in this research, defined as:

$$\eta = \sqrt{|J(q)J(q)'|}, \quad (4.1)$$

where q and $J(q)$ are, respectively, the joint angles and Jacobian matrix of the manipulator. The manipulability metric gives an indication of how far the manipulator is from its kinematic singularities. Kinematic singularities occur when there is a change in the instantaneous degree of freedom of the manipulator. In most cases, kinematic singularities greatly affect the performance and control of a manipulator resulting in control algorithm breakdowns, loss of stiffness or compliance, and intolerable forces or torques on the links. From an objective point of view, losing a degree of freedom means the manipulator cannot further reach or exert force in some direction, and so limit the capabilities of the manipulator. Figure 4.2 shows examples of high and low manipulability. The ability of a nearly fully extended manipulator to reach further outwards is less compared to a more folded manipulator.

For its application in this research, the wrist joints were excluded from the calculations, as done before by Yamamoto [Yamamoto and Yun 1995], and only the base, shoulder and elbow joints are taken into account. This simplifies the manipulability metric to be directly proportional to the sine of the elbow joint angle. This is because an outstretched or completely folded manipulator ($\sin 0^\circ$ or $\sin 180^\circ$) effectively reduces the manipulator DOF by one since it will no longer be able to move in one of its axes. The exclusion of the wrist joints reduces the complexity of the manipulability calculation, without detracting from the measurement of larger movement capabilities that are the focus of this research. Optimal manipulability occurs when $\eta = 1$, and no manipulability when $\eta = 0$.

Due to the nonholonomic mobile base in the mobile manipulator system, the orientation of the mobile base affects the manipulability: it is easier to move in one direction than the other; that is, forwards compared to sideways. Therefore it is ideal for the mobile base to be pointed in the direction of the objective to maximise manipulability. However, to know how to orient the mobile base, the controller requires knowledge of the objectives beforehand. Without this, the preferred orientation for the mobile base is unknown and any arbitrarily imposed preferred orientation is a guess. This research is focusing on developing a reactive controller to improve robustness and reduce the need for planning, therefore the overall objectives of the mobile manipulator are unknown to the controller. Fortunately, unlike the manipulability of the manipulator, which is

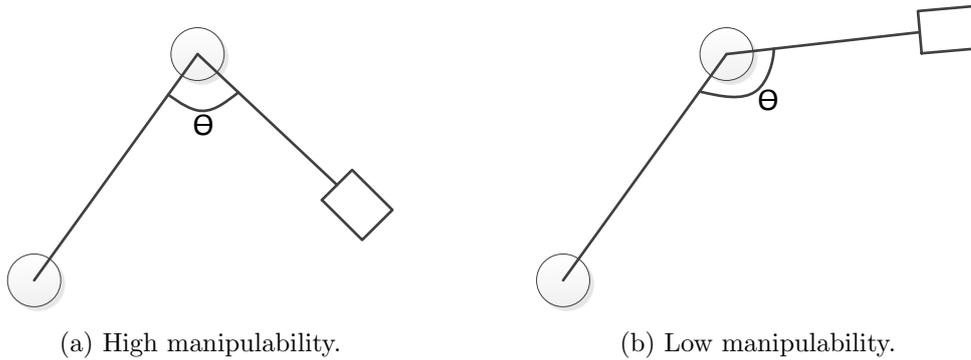


Figure 4.2: Example of high and low manipulability where θ denotes the elbow joint angle.

an indication of how far the manipulator is from its kinematic singularities, having low manipulability with respect to the base does not reduce the degrees of freedom, but only affects how fast or how easily it can move from one direction to the other. Thus calculating the manipulability of the mobile base is neither feasible nor particularly useful in this case. Therefore, the aim of the proposed controller is to measure and control the manipulability of the manipulator in its local coordinates relative to the current location using the mobile base to reposition the manipulator to improve the manipulability.

4.1.2 Stability

The stability metric used for this research is a measure of how close the mobile manipulator system is to tipping over. The main cause of tip-over is the extension of the manipulator beyond the confines of the mobile base. This is amplified if a load is placed at the end effector; for example, if the end effector is gripping on to or grabbing an object. The purpose of the stability metric is to quantify how close the current configuration of the mobile manipulator is to the tip-over point.

Proximity of the tip-over point is calculated from the sum of the moments about each of the axes around which the mobile manipulator can tip over. For the robot model assumed in this research the tip-over axes are along the four edges of the mobile base. As the manipulator is placed at the centre of the mobile base, the tip-over limits are identical for the parallel edges. This reduces the need to calculate the moments about each of the individual tip-over axes. The moment created by the manipulator along the local x and y axes of the mobile base can be instead calculated and these values compared to the maximum counter moments possible. Equations (4.2) and (4.3) calculates the total moment about the x and y axes:

$$M_x = \sum_{l \in Link} m_l g d_{xl}, \quad (4.2)$$

$$M_y = \sum_{l \in \text{Link}} m_l g d_{yl}, \quad (4.3)$$

where

M_x : total moment about the x -axis,

M_y : total moment about the y -axis,

m_l : mass of the link l ,

g : acceleration due to gravity,

d_{xl} : straight line distance between the x -axis and the centre-of-mass of link l ,

d_{yl} : straight line distance between the y -axis and the centre-of-mass of link l .

If the moment, M_x or M_y , exceeds the maximum counter moment then the robot will experience tip-over. This can be visualised with a see-saw where one end has a constant mass representing the maximum counter moment and the other end a variable mass representing the variable moment caused when the manipulator moves. The tipping of the see-saw is analogous to the mobile manipulator system going into a tip-over state. The maximum counter moment available is dependant on the geometry of the mobile base and the total mass of the mobile manipulator system. The centre-of-mass of the mobile base is assumed to be in the middle of the mobile base, so it is equidistant for each of the parallel edges. The maximum counter moment possible about a base edge corresponds to the entire weight force of the mobile manipulator acting relative to that edge. To achieve the maximum counter moment would mean all the reaction force of the mobile manipulator is only acting through one of the parallel edges and the mobile manipulator could balance on the one edge alone. Equations (4.4) and (4.5) calculates the maximum counter moment about the local x and y axes:

$$M_{ct,x} = \frac{W}{2} F_w, \quad (4.4)$$

$$M_{ct,y} = \frac{L}{2} F_w, \quad (4.5)$$

where

$M_{ct,x}$: maximum counter-moment about the x -axis,

$M_{ct,y}$: maximum counter-moment about the y -axis,

W : the width of the mobile base,

L : the length of the mobile base,

F_w : total weight force of the mobile manipulator through the centre-of-mass.

The stability measure about either the x or y axis is then defined as the ratio of the maximum counter moment to the net moment generated by the manipulator about that axis given by:

$$Stability_x = \frac{M_{ct,y}}{M_x}, \quad (4.6)$$

$$Stability_y = \frac{M_{ct,x}}{M_y}. \quad (4.7)$$

Figure 4.3 shows a conceptual model of the stability metric calculation. The beam represent a one-axis view of the mobile base. The manipulator is replaced by its effective moment about the centre of the mobile base, and the total weight force incorporates the mass of both the mobile base and manipulator. F_{R1} and F_{R2} represents the reaction forces along the edges of the mobile base. The sign of the stability metric determines the direction of the net moment with a clockwise direction defined as positive and an anticlockwise direction as negative.

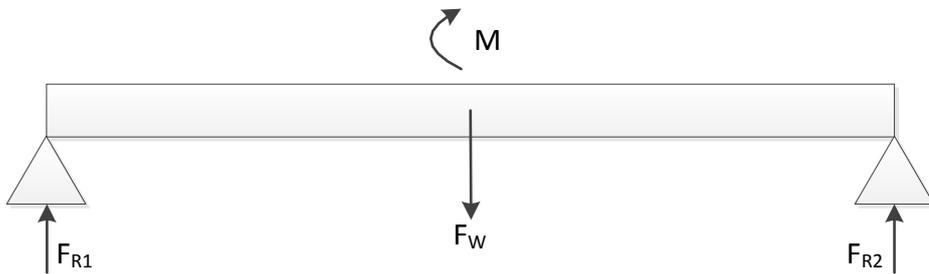


Figure 4.3: Simple free body diagram of a beam representing the calculations used to calculate the stability metric.

The stability metric can also be viewed as a ratio between the distance from the centre of the base to an edge, and the distance from the centre of the base to the current manipulator centre-of-mass. If this ratio is less than 1, then the centre-of-mass lies beyond an edge of the robot, and tip-over will occur. Thus an absolute stability value of greater than 1 is considered stable, with the sign of $Stability_x$ and $Stability_y$ indicating the net direction of the moment about their respective axes.

In the simulation environment used for this research, the implementation of the stability metric was done using a MATLAB function. An excerpt of the main loop for calculating the total moments contributed from the links is shown in listing 4.1. The input packet of the function provides the joint coordinates of the mobile manipulator, and the position and direction of the mobile base. Lines 4 to 8 calculate the coordinates of the centre-of-mass of each link and removes the positional and directional component from the mobile base. This allows the adjusted centre-of-mass coordinates to be used

to directly calculate the moment contributions of each link. This step is necessary as the joint coordinates received by this function are in the global reference frame and the stability metric is also affected by the relative shoulder angle between the manipulator base and the mobile base. Lines 13 to 21 then calculate the moment contribution of each link and sums the total mass of the links and external load used for calculating the maximum counter moment.

Listing 4.1: MATLAB code for calculating the stability metric.

```

1 function [output] = stability(arm,base,ext_load,input_packet)
2 %calculates the moment about the x and y axis.
3 for i = 2:n
4     com_coord(i,:) = (joint_coord(i,:)+joint_coord(i-1,:))/2;
5     temp_coord = com_coord(i,)-pos;
6
7     temp_frame = transl(temp_coord);
8     moment_coord = transl(anti_rot*temp_frame);
9
10    %now the moment coord should all be wrt to the base being
11    %in the original position facing the x axis
12
13    if i==n %apply the load at the last link
14        mx(1,i) = moment_coord(2)*(L{i}.m+ext_load)*g;
15        my(1,i) = moment_coord(1)*(L{i}.m+ext_load)*g;
16        total_mass = total_mass + L{i}.m+ext_load;
17    else
18        mx(1,i) = moment_coord(2)*L{i}.m*g;
19        my(1,i) = moment_coord(1)*L{i}.m*g;
20        total_mass = total_mass + L{i}.m;
21    end
22 end
23 }

```

4.2 HIGH-LEVEL CONTROLLERS

The goal of a high-level controller is to perform a small, specific task required of the mobile manipulator. A collection of high-level controllers can then be used to perform more complex tasks. This approach of creating multiple small, specific controllers and using them as building blocks to achieve a more complex task is based on behaviour-based control, where each behaviour represents an individual high-level controller. The collection of controllers include: objective control, manipulability control, and stability control. The priority of each controller is different, such that should a situation arise where multiple controllers could be used only the highest priority controller is executed. The entire collection of controllers can be seen as a purely reactive controller, with each

of the sub-controllers being behaviours reacting to certain stimuli. Behaviour-based control is a proven method of structuring robotic controllers for unknown and dynamic environments. This provides a more robust controller which does not rely on any form of global path planning to perform its task but a set of specialised sub-controllers to tackle issues as they arise.

Using a behaviour-based approach also addresses the modularity design goal of the overall controller. Behaviours in behaviour-based control are inherently modular as they encapsulate how to perform a specific task without the need to use other behaviours in the same complexity level. Having a modular controller also implies the ability to add, change or remove controllers without significant effects on the rest of the controllers. This is reinforced in the design implementation of the individual high-level controllers described in the following sections by ensuring all the controllers have access to the same resources with a standard interface between the different levels in the overall controller.

4.2.1 Objective Controller

The goal of the objective controller is to control how the mobile manipulator approaches and reaches a target point with its end effector. The main decision performed in this controller is whether the target point is within reach of the manipulator or out of range. The range limit is a freely set variable for the objective controller but is constrained on the upper limit by the length of the links in the manipulator. If the target point is out of range, the controller sets the base to move closer to the target until it is in range of the manipulator. Being in range does not necessarily mean the manipulator can reach the target, as the ability to reach the target depends on the valid kinematic configurations capable of reaching the target point. Inverse kinematics are calculated for the manipulator to reach the target point and if a solution is found, it is then executed. If no inverse kinematic solution is found the base is set to move closer to the target point. A default pose which maximises manipulability and keeps stability above the threshold is maintained during a base only move operation.

Listing 4.2 shows the MATLAB code used to implement the objective controller. The main conditional decides if the mobile manipulator is within range of the target. The inverse kinematics for the manipulator is calculated on line 7, and, if successful, the result is modified to take into account the effects of base direction. A bounding function is applied to limit the result to within -180° and 180° on lines 13 to 14. This is to ensure the each joint travels the shortest angular distance to reach their targeted joint angle. If the inverse kinematics calculation returns no solution the base is set to move closer to the target.

Listing 4.2: MATLAB code for the objective controller.

```

1 if dist_to_target > arm_range
2     base_state = base_target;
3     arm_state = arm_default;
4     current_state = Mbase;
5 else
6     try
7         ikine_result = ikine560(arm, ikine_target, 'ru');
8         ikine = true;
9     catch err
10        ikine = false;
11    end
12    if ikine
13        ikine_result(1) = ikine_result(1) - dir*pi/180;
14        ikine_result = number_wrapper(ikine_result, pi);
15        base_state = base_stop;
16        arm_state = arm_target;
17        current_state = Marm;
18    else
19        fprintf('Cannot reach point\n');
20        base_state = base_target;
21        arm_state = arm_default;
22        current_state = Mbase;
23    end
24 end

```

4.2.2 Manipulability Controller

The goal of the manipulability controller is to improve the manipulability of the mobile manipulator during task execution. The manipulability of the manipulator solely depends on the angle of the elbow joint. Therefore the only meaningful way to improve the manipulability is to move the base while still attempting to reach the same target point. This means driving towards the target if the manipulator is outstretched, or away from the target if the manipulator is folded. Obviously instead of moving the base the manipulator could simply move itself to maximise the manipulability, but this would mean moving the end effector away from the target point. Four quadrants are defined around the base to determine the direction to move the base to produce the fastest improvement in manipulability, that is, the fastest direction of travel to minimise the distance between the base and current location of the end effector of the manipulator. As illustrated in Figure 4.4, these quadrants each cover a 90° range that is $\pm 45^\circ$ of the robot axes. Figure 4.4 also depicts the direction of motion for the base to move in should the manipulator fall into one of the quadrants for a manipulator that

is outstretched. For a manipulator that is folded the opposite directions are used.

Listing 4.3 shows the main conditional to decide which manipulability movement sector to execute. The range variables used in the conditionals set up the boundaries of the sectors, and the check to see if a manipulator is folded or outstretched is performed in lines 1 to 3. Arbitrary coordinates are then created, depending on which manipulability sector is executed, to the front, back, left or right of the mobile base for the low-level base movement controller to execute.

Listing 4.3: MATLAB code for the manipulability controller.

```

1  if abs(joint_angles(3)) > pi/2
2      correction_dir = -1;
3  end
4  if(joint_angles(1)>fwd_range && joint_angles(1)<fwd_range2)
5      %outstretched in the forward direction, move base forward
6      adjustment_coord=sca2coord([adj_amount dir*pi/180]);
7      base_target=current_coord+adjustment_coord*correction_dir;
8  elseif (joint_angles(1)>l_range && joint_angles(1)<l_range2)
9      %outstretched in the left direction, move base left
10     adjustment_coord=sca2coord([adj_amount (dir+90)*pi/180]);
11     base_target=current_coord+adjustment_coord*correction_dir;
12 elseif (joint_angles(1)>r_range && joint_angles(1)<r_range2)
13     %outstretched in the right direction, move base right
14     adjustment_coord=sca2coord([adj_amount (dir-89)*pi/180]);
15     base_target=current_coord+adjustment_coord*correction_dir;
16 elseif (joint_angles(1)>b_range || joint_angles(1)<b_range2)
17     %outstretched in the rear direction, move base backward
18     adjustment_coord=sca2coord([adj_amount (dir+180)*pi/180]);
19     base_target=current_coord+adjustment_coord*correction_dir;
20 end

```

The threshold that the controller must keep the manipulability performance metric above is a “soft threshold”. This means that although the manipulability controller is activated and the base starts to move when this threshold is reached, it does not stop the manipulator from attempting to reach the target point. This can result in the manipulability temporarily dropping below the threshold as the manipulator movement can be faster than the base readjustment movement in the required direction. A soft threshold was chosen because the consequences of temporarily not maintaining the manipulability of the mobile manipulator are not critical to its operation, while a gain in performance in terms of time to reach the target point can be obtained.

4.2.3 Stability Controller

The goal of the stability controller is to prevent the robot from tipping over. The stability metric is used to measure proximity to tip-over and the stability controller is

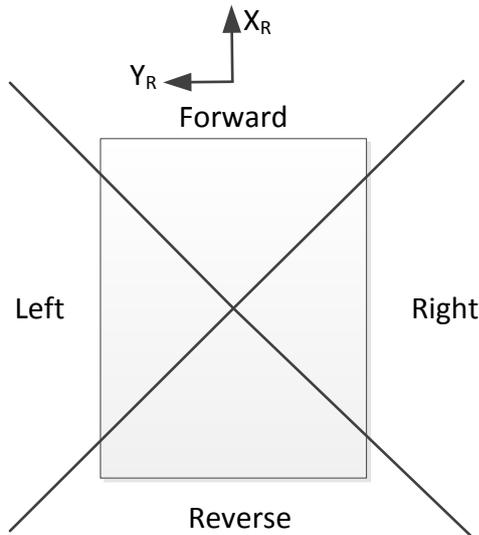


Figure 4.4: Manipulability movement sectors for outstretched manipulator.

used to maintain the measured stability within the set thresholds. Tip-over stability is a critical aspect in the operation of a mobile manipulator and a common issue with mobile manipulation. The simulated robot in this research can become unstable and tip over in either of the two local axes, x and y . There are two options to improve the stability metric:

1. The distance between the base and the manipulator links can be reduced, thus decreasing the total moment imparted on the base.
2. The maximum counter moment for an axis can be increased. This can only be done if the geometric shape of the base is different around each of the two axes.

In either case, again, just like the manipulability controller, the only meaningful option is to move the base and maintain the current position of the end effector. Because of the nonholonomic nature of the base, direct movements to the left and right of the base are not possible, therefore the methods used in controlling the stability of the robot are slightly different depending on which axis is approaching the stability threshold. There are two cases of controlling for stability, one for each axis:

1. The stability metric is approaching the threshold in the y -axis. This means that the robot is approaching tip-over about the y -axis and the manipulator is predominantly outstretched either in front or behind the base, depending on the sign of the stability metric.
2. The stability metric is approaching the threshold in the x -axis. This means that the robot is approaching tip-over about the x -axis and the manipulator is

predominantly outstretched either to the left or right of the base, depending on the sign of the stability metric.

The solution to each of these cases is outlined below:

1. *y*-axis: the base must drive forwards or backwards while the end effector is maintained in the same global position. This results in a shortening of the distance between the end effector and base, thus reducing the moment load about the *y*-axis.
2. *x*-axis: the rectangular shape of the mobile base is exploited to optimise this controller. Instead of directing the base to move directly to the left or right to reduce the moment load, which is difficult to do due to the nonholonomic constraints, the base is rotated to increase the maximum counter moment and so increase the stability metric. The benefit of this approach is that base movements are potentially reduced, saving time and energy. Due to the rotation, the axis of concern for the stability metric has been transformed to be in the *y*-axis and any further improvements can be performed using the *y*-axis solution.

Figure 4.5 summarises the actions taken by the controller described previously.

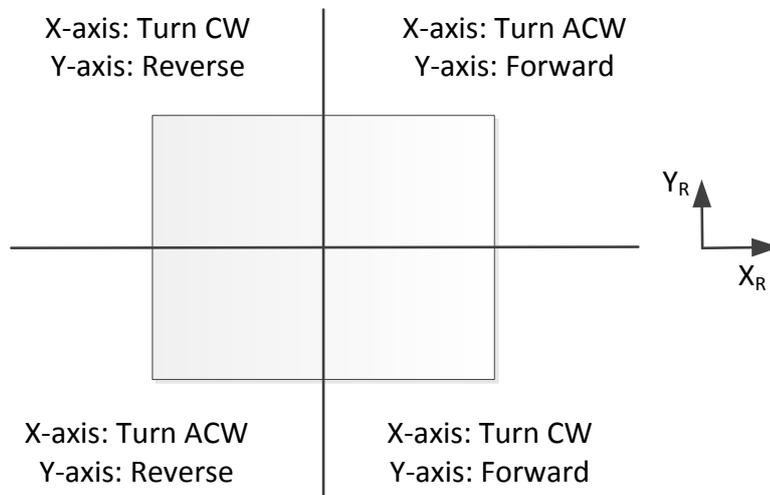


Figure 4.5: Stability quadrants and corresponding controller responses.

Listing 4.4 shows the MATLAB implementation of the stability controller. Lines 2 to 11 address the stability in the *x*-axis by rotating the mobile base towards the end effector and lines 14 to 19 addresses the *y*-axis stability adjustment by driving the mobile base towards the end effector. The direction adjustment in lines 5 to 8 is required when the manipulator is outstretched behind the mobile manipulator and rotational directions must be reversed to ensure the base reaches the goal of facing the end effector in the shortest movement. Facing the end effector in this case does not mean being in front of the mobile base, just that it is aligned with the *x*-axis of the

mobile base as driving in reverse is considered to perform equally compared to driving forward.

Listing 4.4: MATLAB code for the stability controller.

```

1  %X-axis
2  if abs(stab_measure(1)) < stab_limit %tipped over
3      unstable = true;
4  elseif abs(stab_measure(1)) < current_stab_boundary
5      if abs(joint_angles(1)) >= pi/2
6          dir_adjust = -1;
7      else
8          dir_adjust = 1;
9      end
10     change_dir = sign(stab_measure(1))*adj_angle*dir_adjust;
11 end
12
13 %Y-axis
14 if abs(stab_measure(2)) < stab_limit %tipped over
15     unstable = true;
16 elseif abs(stab_measure(2)) < current_stab_boundary
17     adj_coord = sca2coord([adj_amount dir*pi/180]);
18     base_target=current_coord+sign(stab_measure(2))*adj_coord;
19 end

```

4.3 CONTROLLER STATE MACHINE

The goal of the controller state machine is to provide high-level coordination of the different behaviours. A state machine approach was selected for its simplicity and practicality, and also because it has been used in successful behaviour-based robots [Arkin 1998]. The state machine approach is flexible, in that new states can be added as required, or transitions altered to change the behaviour of the overall controller. A tradeoff in using the state machine approach is the increase in complexity should the amount of states increase. However the modular approach that has taken in the overall controller design should allow the state machine to be replaced with another more suitable structure with minimal changes to the rest of the controller.

The state machine controls state transitions based on a set of threshold values and the overall goal of the mobile manipulator. The main states the robot may be in are:

- Go to target states: attempt to reach the target way point with the end effector by executing the objective controller. Divided into move base and move manipulator states.
- Address manipulability state: improve the manipulability if the manipulability threshold is crossed by executing the manipulability controller.

- Address stability state: improve the stability if the stability threshold is crossed by executing the stability controller.

Two general conditions are used to see if a state transition will occur:

1. Is a measurement above or below certain thresholds?
2. Are there any other states more important than this one that need to take priority?

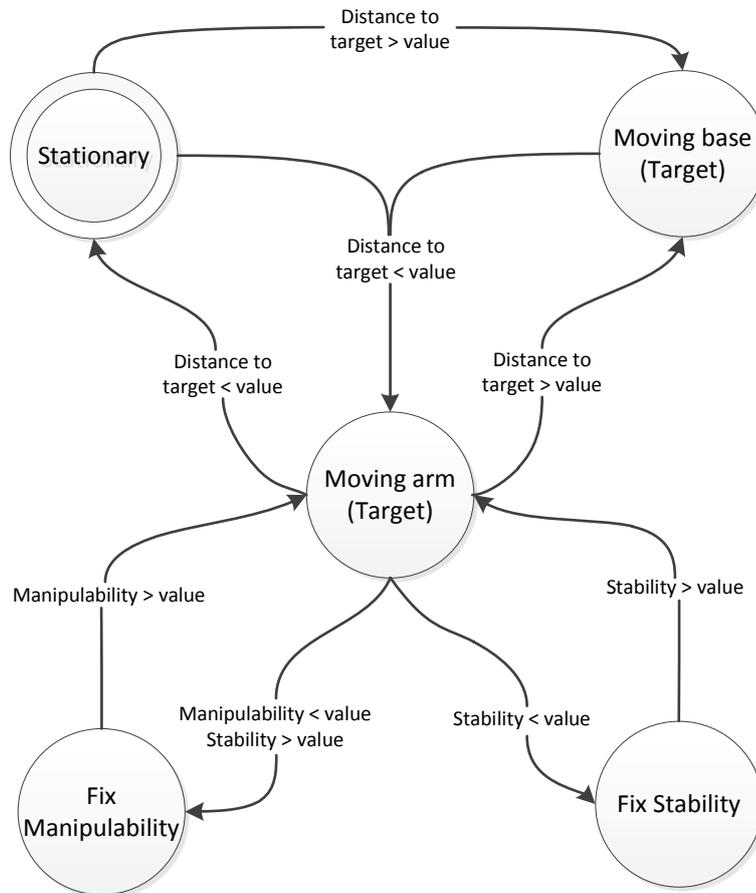


Figure 4.6: State diagram of the controller and the conditions that cause transitions.

Figure 4.6 illustrates the states, and transitions from one state to another. In the current configuration, stability has the highest priority. There are two “moving to target” states that indicate whether the base or manipulator is used to reach the target way point. The moving-base state is entered if the target is out of range of the manipulator. Once the range threshold is crossed, the moving-base state transitions to the moving-arm state. The threshold to determine if the target is in range of manipulator is set to be 80% of the maximum arm range. Once the base is within the defined range, a check is performed to see if a valid solution for the manipulator to reach the target point is possible from the inverse kinematics. If both of these

conditions are satisfied then there will be a state transition from the moving-base state to the moving-arm state. If there are no valid solutions for the manipulator then the base remains in the moving-base state and creeps closer towards the target. The 80% value was empirically selected to increase the chance that a solution to the inverse kinematics can be found on the first attempt.

To enter the fix-stability state requires the stability metric about any of the measured stability axes to be below a set threshold. The stability threshold is set to 1.25 to provide a safety margin in case certain actions in the process of improving stability temporarily makes it worse. To transition out of the fix-stability state a higher stability threshold of 1.6 is to be achieved before normal operation can resume. This hysteresis significantly reduces the number of state oscillations between the fix-stability state and the moving-arm-target state. The hysteresis is also desirable in the physical world, as it reduces the amount of stop-starts associated with moving the base and arm to fix stability and then moving the arm to target again. The physical effects of many stop-starts can include unwanted vibrations, and energy wastage due to having to constantly restart an electric motor.

To enter the fix-manipulability state, the manipulability must drop below 0.75. To transition out of the fix-manipulability state, a manipulability of 0.9 must be achieved. This difference in thresholds is the same hysteresis idea applied in the stability state transitions to reduce state oscillations. As the controller have been configured to prioritise stability higher than manipulability, the robot can also exit the fix-manipulability state if the stability metric drops below the stability threshold.

4.4 LOW-LEVEL CONTROLLERS

Two low-level controllers, base movement and manipulator movement, are responsible for the actual movement of the mobile manipulator. These controllers are at the bottom of the hierarchical tree, and are shared by all of the different behaviour modules. The competitive coordination method is used by the overall controller, which means that even though all the behaviour modules have access to the low-level controllers, the low-level controllers will only execute the movements of one of the behaviour modules as determined by the controller state machine. The following sections detail the design of the low-level controllers.

4.4.1 Base Movement Controller

The base movement controller uses a proportional-derivative (PD) controller design. A PD controller was selected for its ease of comprehension and implementation, and the fact that it does not require an accurate system model to perform well. These characteristics also makes PD controllers common in industry. However, the modular

nature of the overall controller architecture makes it straightforward to replace the PD controller with something more sophisticated if that proves necessary or advantageous.

The base movement controller is divided into two parts, one responsible for forward and backward motions, and the other responsible for rotation. The design of both of these parts is identical. Table 4.1 shows the input and outputs of the controller. The error input when used for the forward motion controller is the straight line distance between the mobile base and the target point in the ground plane. The error input when used for rotational control is the difference in angle from the current heading of the mobile base to the heading of the target point. The distance error and direction error are illustrated in figure 4.7 and a block diagram of the controller is shown in figure 4.8. A simple brake controller is also implemented, which again, uses a PD controller to control the velocity of the mobile base to zero, using the current translational and angular velocities as inputs.

Table 4.1: Inputs and outputs of the base movement controller.

Input	Output
Error	Force left side Force right side

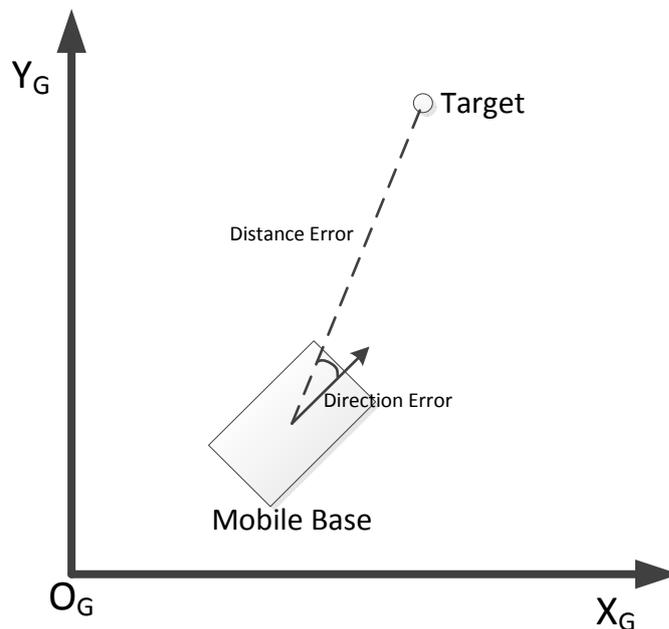


Figure 4.7: Error definitions for mobile base movement control.

The “force left side” and “force right side” outputs are calculated individually for the translation and rotation controller and represent the left and right drives of the mobile base. The left and right side force outputs of the respective controllers are then combined together to create the final output. To calculate the left and right side force outputs of the translation PD controller, the controller first calculates the total force

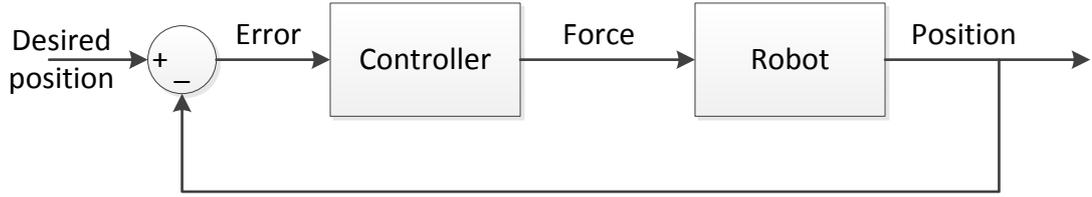


Figure 4.8: Block diagram of the PD controller.

based on the distance error. The total force is then halved to produce the left and right side force outputs. A similar process is performed by the rotation PD controller, the total force to generate the torque is calculated based on the directional error, and then halved as before to produce the left and right side force outputs. However, a sign change is then applied to the left side force output to create the force imbalance that will induce turning. The choice of sign change on the left side force output is related to the sign definitions of the model, namely, a positive torque is in the anti-clockwise direction. Equations (4.8) and (4.9) show the calculation of the final left and right side force outputs.

$$force_left_side = F_{translation}/2 - F_{rotation}/2 \quad (4.8)$$

$$force_right_side = F_{translation}/2 + F_{rotation}/2 \quad (4.9)$$

The gains used for the PD controllers are shown in table 4.2 and were obtained empirically. The gains were selected to satisfy a performance criterion of being able to move within a reasonable time, defined as a rise time of 3 s for small errors, and to eliminate overshoot. Steady state errors are not a big concern as an advantage of the mobile manipulator system is that the manipulator can always compensate for base positioning errors.

Table 4.2: Gains used for the PD controllers.

	Translation	Rotation	Brake
Proportional gain	8	0.6	100
Derivative gain	150	1	1

4.4.2 Manipulator Movement Controller

The manipulator movement controller is a pseudo controller that outputs the expected result of applying a real controller to the manipulator model. This result is then taken as the actual position of the manipulator and used throughout the rest of the controllers. By doing this, the dynamics of the manipulator model has been effectively

removed, and are instead simulating the position of the manipulator and providing it to the rest of the controller. From a high-level controller point of view, all it sees is this is the current position of manipulator, and it needs to move to another position to perform a task. Two speed limitations are imposed on the pseudo controller such that the simulated output resembles the behaviour and output of a model controlled with a real controller:

1. A maximum joint speed of 2 Rads^{-1} or about 19 RPM, is allowed for each joint, a fairly conservative limit.
2. The actual joint speed is proportional to the ratio of the joint error of the current joint and the maximum error of any of the joints as shown in equation 4.10 . This also ensures all joints reach their target at the same time.

$$joint_speed_n = \frac{joint_error_n}{\max(joint_error_{n=1..j})} \times max_joint_speed \quad (4.10)$$

The pseudo controller approach is taken to alleviate simulation difficulties. Initially, a PID joint torque controller was built using the provided controller blocks in the Robotics Toolbox. By itself, it was working, but was rather slow. When the manipulator and mobile base was combined together, the solver and step size for the simulation environment had to be adjusted and changed to enable simulation of the integrated model. However, this introduced errors for the manipulator controller during some scenarios. Simulation speed of the combined system was also very slow, which made testing difficult.

Table 4.3 shows the input and output of the controller and equation (4.11) shows how a resultant joint angle is calculated.

Table 4.3: Input output of the manipulator movement controller

Input	Output
Target joint angle	Output joint angle
Current joint angle	

$$output_joint_angle_n = current_joint_angle_n + joint_speed_n \times time_step \quad (4.11)$$

4.5 INTEGRATED SYSTEM

In the previous sections, the main components that makes up the controller design have been described:

- Low-level controllers responsible for movement of the mobile manipulator,
- High-level controllers responsible for deciding how to achieve the current aim,
- State machine for controller coordination,

as well as the key performance metrics of manipulability and stability. An overview of how all of these parts fit together, along with how the controller interacts with the mobile manipulator model to complete the controller-model system is discussed below.

Generic inputs into the controller system contain positioning and kinematic data of all the mobile manipulator parts. These inputs are made available to all the high-level controllers. Specific inputs into the controller system are the performance metrics of manipulability and stability, and the current task for the mobile manipulator. These specific inputs are routed to the respective high-level controllers. The operation of all the high-level controllers can be separated into two modes, on or off. The outputs of the high-level controllers are all of the same type, that is, new positions for the mobile manipulator to move to. The low-level controller takes the new positioning input and executes the movement. Whether a controller is on or off is dependent on the inputs for that controller; for example, the manipulability controller is on when the manipulability metric is above the manipulability threshold. The controller state machine monitors which high-level controller is on, and if only one is on, the outputs of that controller is passed to the low-level controllers to execute. If multiple controllers are on, the controller state machine decides, based on the defined state transition requirements, which high-level controllers output is passed to the low-level controllers.

The interface between the controller and mobile manipulator model consists of the position and kinematic data of the model, which make up the generic input for the high-level controller, and the outputs of the low-level controllers, which provide the force inputs to model. An overview of the complete Simulink system that connects model and controller, as well as other various data collecting and monitoring tools of the entire simulation, is shown in figure 4.9.

At this stage, a working controller has been implemented capable of performing the task of reaching waypoints in a workspace while maintaining manipulability and stability above a set threshold. In Section 4.6, the process of extending the features of the already functional controller is described by adding an obstacle avoidance controller. The purpose of this is to demonstrate the modularity of the controller design.

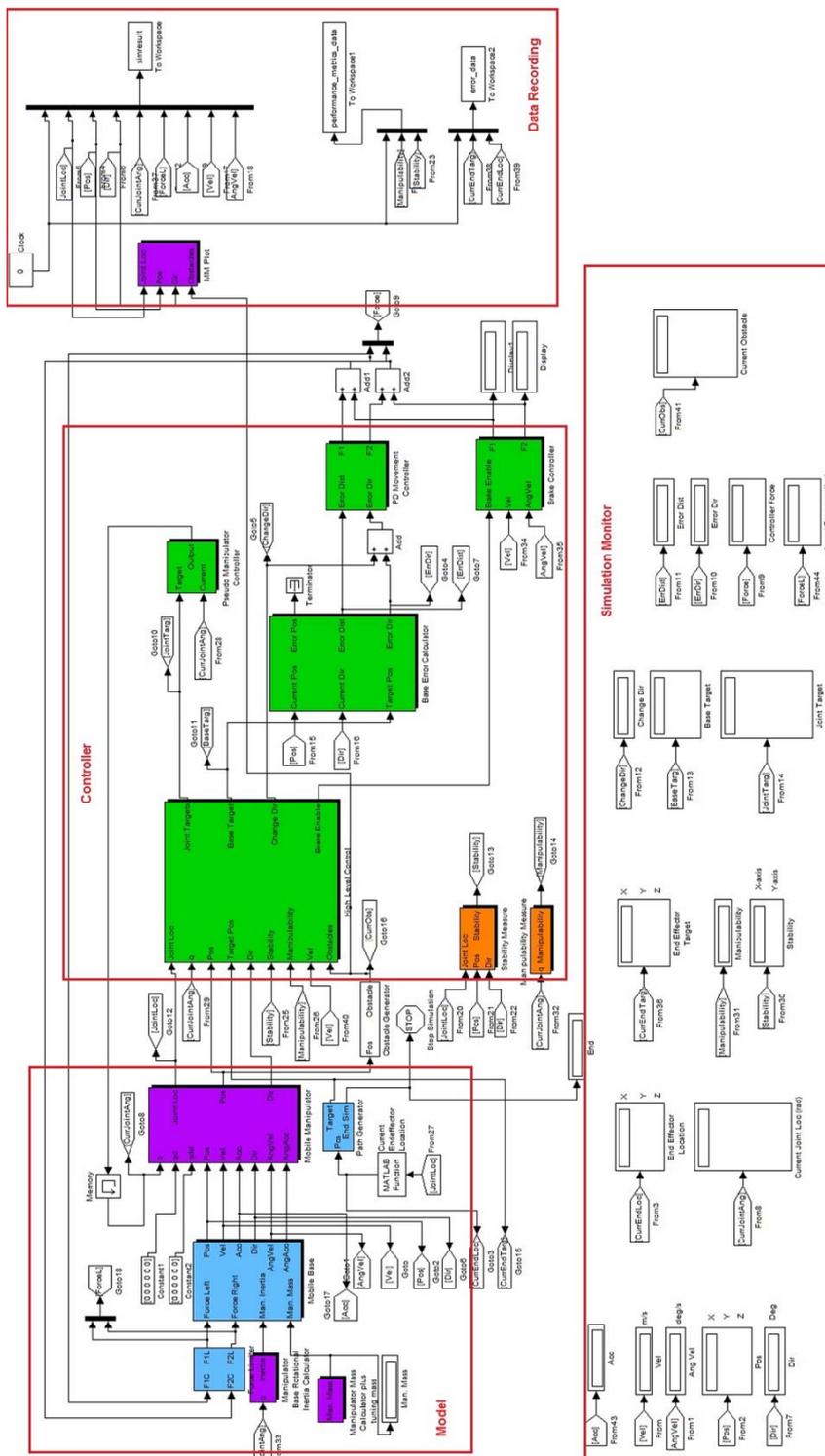


Figure 4.9: Simulink implementation of the controllers connected with the models and other simulation tools.

4.6 EXTENDING THE CONTROLLER

Modularity is an important feature of the controller design. It makes individual aspects of the controller easier to understand, and allows the controller to be adapted to perform new tasks or add new features without the need to redesign the entire controller. The modularity of the controller applies to all levels in the hierarchy, from the arbitrator down to the low-level movement controllers. A different arbitrator can be substituted to replace the state machine, additional high-level controllers can be added to increase functionality, and the base movement PD controllers can be replaced to achieve new performance characteristics. Having modularity alone is less useful if it is very difficult or complicated to create the modules themselves to fit the overall system, therefore the aim is to also minimise the complexity required of the individual controllers in order for them to be compatible with the overall system.

The benefits of the modular controller design is demonstrated in this section by showing how another behaviour, in the form of an obstacle avoidance controller for the mobile base, is incorporated to the already established controller. Obstacle avoidance was selected as it adds another form of movement that needs to be coordinated with the rest of the controllers, and also because it is a common task in the field of robotics. The same principles used to integrate the obstacle avoidance controller into the system can be used to add other behaviours as well. The general principles applied in designing a controller to be integrated into the overall controller is described, as well as the changes required of the system to incorporate it. An obstacle avoidance controller example is then used to illustrate those principles.

4.6.1 Modular Controller Requirements

To maintain the modularity of the system, all high and low-level controllers are designed using the following principles:

- All controllers have access to a common set of inputs for that level through a standard interface.
- Individual controllers can also have specific inputs for that controller. However that input is not available to other controllers.
- Outputs of all controllers are in the same format for that level.
- Controllers in the same level cannot interact with one another. It is the job of the controller in the higher level to provide the coordination.

The high-level controller layer of the system provides the main functionality of the mobile manipulator. This is the level where controller modularity will be most beneficial,

as new behaviours are added in this level. Therefore, the above principles is described in more detail with respect to designing a high-level controller.

The common set of inputs available for all high-level controllers consists of the current global position of the mobile manipulator and the physical characteristics of the mobile manipulator. These inputs were selected as they will most likely be used by any given controller. A balance is required in deciding how many inputs to provide for every controller. If too few inputs are provided, then the controller itself will have to compute the relevant data, increasing the controller's complexity and performing duplicative work if this data is already computed somewhere else. However, too many inputs will inevitably result in a controller having to sort and discard some data that it does not use. More inputs also increases the coupling between the controllers and the overall system, and thus reducing the modularity of the system. Any changes in the common inputs will filter through all the controllers and they will all have to be updated.

Specific high-level controllers may need additional inputs to function; for example, obstacle size and location for an obstacle avoidance controller, and target location for the target controller. Any specific inputs can only be used for that particular controller and should not be used in other high-level controllers. This is done to limit the coupling between different high-level controllers and specific inputs. The idea behind specific inputs is to closely couple a controller to a particular input creating a controller-sensor pair. The controller-sensor pair is then viewed as the module, and should one want to remove functionality from the overall system then one can simply remove the controller-sensor pair without it affecting other controllers.

The outputs of the high-level controllers are in the form of global xy positions for the mobile base to travel to, and joint angles for the manipulator to reach. This is then used by the low-level controllers to execute the movement. The same rationale in selecting the number and type of common inputs for the controllers was applied to determine the outputs. On a basic level, a global position and manipulator joint angles is all that is required to move the mobile manipulator from its current position to the target. Any controller that needs to follow a specific path from the current position to target position can compute the intermediate target positions for the mobile manipulator to follow. Again, this reduces the complexity required of the low-level movement controllers to having to process paths as well as not requiring all the high-level controllers to accommodate path following in their outputs if they do not need it.

High-level controllers cannot directly execute other high-level controllers. This rule is specifically aimed at eliminating direct controller coupling. Figure 4.10 shows the dataflow between controllers and sensors and illustrates the design rules that have been implemented for modularity. The controller state machine controls the switch to

determine which data from the high-level controllers are sent to the low-level controller.

Integration of any new controller to the overall system is mostly performed in the next higher controller layer. In the example of integrating a new high-level controller into the overall system, a new state is created in the state controller to activate the controller. Once the system enters into that state, the controller is executed. State transitions are also required to define the conditions for entering and leaving that state.

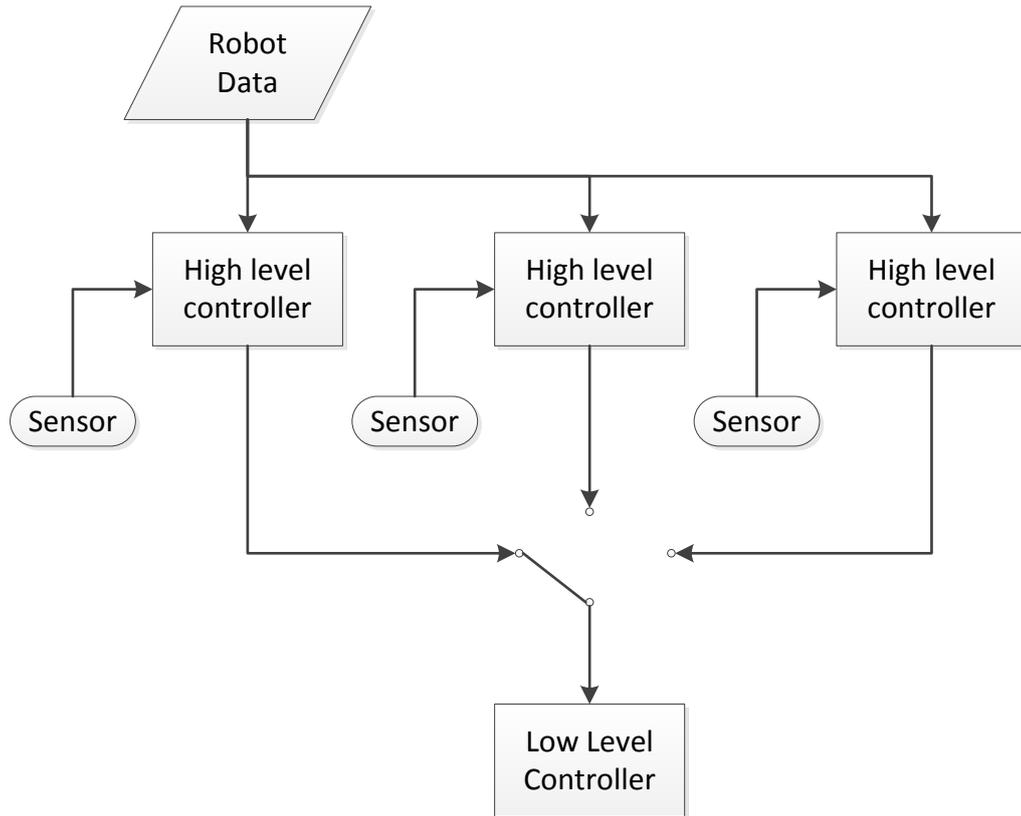


Figure 4.10: Data flow for the overall controller between the different components.

4.6.2 Obstacle Avoidance Controller

As an example of adding a new controller, an obstacle avoidance controller is implemented and added to the existing controller. There are two purposes to the obstacle avoidance controller. One, as the name suggests, is to perform obstacle avoidance to allow the mobile manipulator to manoeuvre around an obstacle during task execution. The second purpose is to demonstrate the modularity of the overall system.

To simplify the obstacle avoidance controller design, limitations and assumptions were made for the obstacle avoidance problem. This is justifiable in this case as the main goal is to demonstrate the modularity of the controller design, not the complexity of the obstacle avoidance. All the obstacles are assumed to be uniformly shaped circular

columns, and that perfect knowledge of obstacles and robot position is available. A circular area is also defined that is just large enough to enclose the mobile base and this is used as the mobile base model for obstacle avoidance. This simplifies the obstacle avoidance problem to be two dimensional, with a goal of preventing the two circles, the obstacle and the mobile base, from colliding. Using circles makes collision detection easy: the distance between the centres of the circles can be compared to the sum of the radii of both circles. Should the distance be less than the sum of the two radii, then a collision is registered. A buffer distance between the circles representing the mobile base and obstacle is defined, such that once the mobile base enters the buffer zone, obstacle avoidance is performed. The minimum size of the buffer zone can be adjusted to reflect how close the mobile base should be allowed to approach obstacles. The maximum size of the buffer zone is dynamically set and is proportionally related to the velocity of the mobile base. Therefore, a faster moving mobile base enters the obstacle avoidance state sooner. The buffer size is determined with the manoeuvrability of the mobile base and its controller performance in mind, as within the buffer distance the mobile base must be able to steer around the obstacle.

For the obstacle avoidance controller to fit within the already established control structure, the obstacle avoidance controller must output destination points in the global reference frame. These destination points can then be used by the existing base-movement controller to move the mobile base. The basic obstacle avoidance strategy is to have the obstacle avoidance controller output a series of temporary waypoints for the mobile base to follow until the obstacle is cleared, after which execution of other tasks can resume. The inputs used for the obstacle avoidance controller are the current mobile base location and current obstacle location. Table 4.4 shows the input and output of the obstacle avoidance controller.

Table 4.4: Inputs and outputs of the obstacle avoidance controller.

Input	Output
Mobile base location Obstacle location and size	Temporary waypoint to move

The presence of an obstacle is only signalled to the controller once it is close to the mobile base, to reinforce the reactive nature of the controller design and eliminating the possibility of performing any path planning. Once an obstacle is presented to the controller and the controller is activated, the first check is to see if the obstacle is actually in the way. If the obstacle is close to the mobile base, but the next path for the mobile base is away from the obstacle, then no actions are required. The check to see if an obstacle is in the way is done by comparing the angle between a line connecting the centroids of the base and obstacle to the heading of the next goal point. If the difference in angle differs substantially, then the obstacle is considered “not in the way”.

The core of the obstacle avoidance controller is the calculation of the temporary

waypoints that take the mobile base around the obstacle. A temporary waypoint is calculated based on the current vector between the mobile base and obstacle. The waypoint is placed with a constant offset along a vector perpendicular to the robot-obstacle vector and tangent to the circle enclosing the mobile base. The direction of the robot-obstacle vector changes as the mobile base drives around the obstacle, and a new temporary waypoint is recalculated as it moves. Thus the mobile base traces out a circular path around the obstacle until the obstacle is no longer in its way using the previously discussed method. Figure 4.11 shows a typical obstacle avoidance scenario, where the temporary waypoints—indicated by crosses—are shown to be computed as the mobile manipulator drives around the obstacle.

Listings 4.5 show an excerpt of the obstacle avoidance algorithm. Lines 2 to 3 perform the check to see if an obstacle is in range and lines 8 to 25 perform the actual obstacle avoidance. House keeping operations include adjusting for sign changes depending on the location obstacle relative to the mobile base and its direction of travel.

Listing 4.5: MATLAB code for performing obstacle avoidance.

```

1  if obstacle_state ~= 0 && base_state == base_target_state
2      dist_obs=sqrt(sum((current_coord(1,1:2)-obs(1,1:2)).^2));
3      if dist_obs<obs_clr+abs(velocity)*0.7+base_bubble + obs(4)
4          %
5          %Lots of house keeping operations
6          %
7          %check if obstacle is in the way
8          if (abs(theta_obs - theta_targ)) > pi/3
9              obstacle_state = obstacle_base_clear_state;
10             else %need to start to adjust base to drive around
11                 obstacle_state = obstacle_base_range_state;
12                 new = zeros(1,3);
13                 %vector from current pos to obs
14                 new_(1,1:2) = (obs(1,1:2) - cur_coord(1,1:2));
15                 %rotate it so it is flat, easier to work with
16                 new = new*rotz(theta_obs);
17                 %getting new x coordinate to go to
18                 new(1) = new(1) - obs(4) - obs_clr;
19                 %new y target is 2.5 times obstacle radius
20                 %perpendicular to outer radius of base
21                 new(2)=new(2)+dir_adj*b_adj*obs(4)*2.5;
22                 %rotate it back to actual coordiante system
23                 new = new*rotz(-theta_obs);
24                 new = new + cur_coord;
25             end
26         end
27     end

```

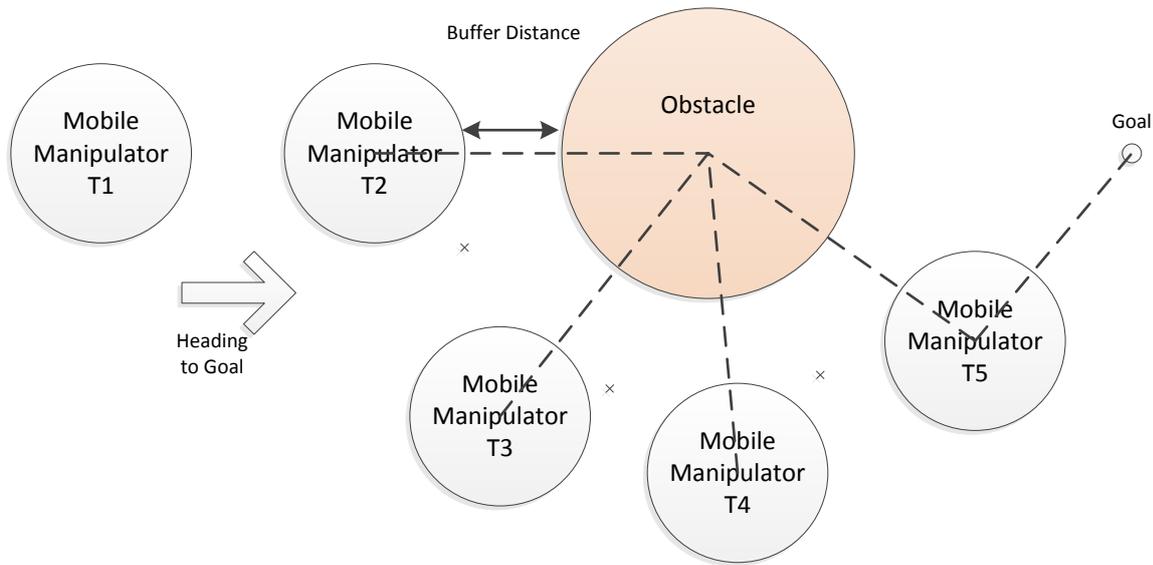


Figure 4.11: Example of the steps in the operation of an obstacle avoidance sequence.

4.6.3 Controller Integration

The controller state machine coordinates and decides when to transition from one high-level controller to another. To integrate the obstacle avoidance controller, a new obstacle avoidance state is defined, along with conditions to enter and leave the new state, as shown in figure 4.12. To enter the obstacle avoidance state, the mobile base has to have breached the buffer distance, and also be heading towards an obstacle. To leave the state the mobile base must be out of range of the buffer distance or be heading away from the obstacle. Manipulator movement is frozen during the obstacle avoidance state so that stability and manipulability metrics are unchanged during the obstacle avoidance process. The buffer zone between the mobile base and any obstacle is assumed to be large enough to ensure collisions do not occur when the mobile base repositions to improve manipulability. This particular situation can arise when a goal point is close to an obstacle but the obstacle is not in the way. However, additional transitions from the obstacle avoidance state and the other states could be added if necessary.

Integration of the obstacle avoidance controller does not require changes to any of the other high-level controllers, and does not interact with or rely on them to function. It makes use of the existing low-level movement controllers just like the other high-level controllers. Once integrated, the obstacle avoidance controller becomes just another high-level controller that the controller state machine selects based on the current feedback from the surroundings.

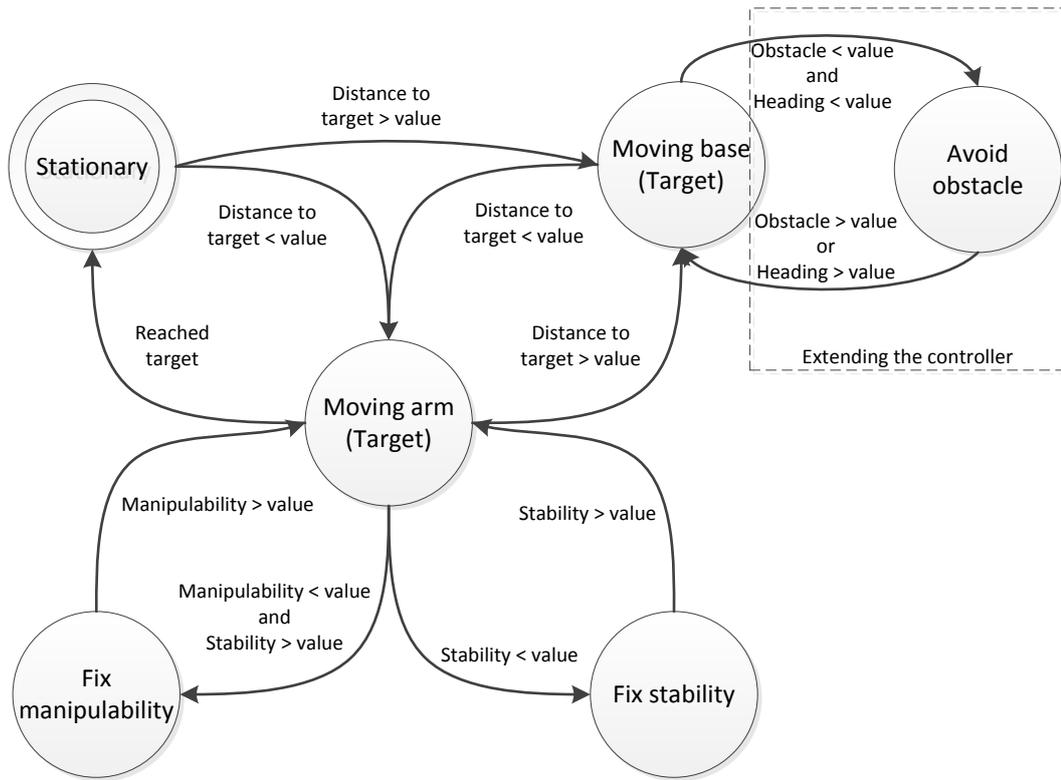


Figure 4.12: Updated controller state machine with obstacle avoidance.

4.7 SUMMARY

The performance metrics of manipulability and stability have been defined, along with how they are calculated and how they are used by the controllers. A behaviour-based approach is used to create three high-level controllers to perform task execution, manipulability control and stability control using the aforementioned performance metrics. The same behaviour-based approach is used to create low-level controllers responsible for the movement of the mobile base and manipulator. Coordination of all the controllers is achieved through a state machine controller, which determines what controllers to execute based on the current operating status of the mobile manipulator. The process to add additional controllers, and the ease in how it can be done due to the modular controller design is also described. To demonstrate the modular controller design, a simple obstacle avoidance controller is created and added to operate with the existing controller.

In Chapter 5, the simulation setup that will be used to test the controller to evaluate its performance is outlined. A range of performance metric values will be used to better understand what, and how much they affect the performance of the controller.

Chapter 5

SIMULATION RESULTS

To demonstrate and characterise the operation of the controller, an investigation into how the controller performs when tackling arbitrary tasks is required. The tasks that the controller will be tested on in the simulations for this research will involve reaching a series of waypoints with the end effector. Although testing the controller on a few example scenarios that can show the controller can work in some situations, it does not provide much confidence that the controller is able to operate for arbitrary tasks. To gain greater confidence, the controller is instead tested on a large set of randomly generated scenarios, using a range of performance metric thresholds to see how well the controller performs under all these situations. All the simulations are performed in MATLAB Simulink.

In the first section of this chapter, the simulation setup and the type of data that will be collected to evaluate the performance of the controller is described. The simulation results of a baseline configuration of the controller is then presented, which consists of average threshold values for both the manipulability and stability. The threshold values of manipulability and stability are then varied, and the results compared with the baseline simulation. The effects of the thresholds on the performance of the controller can then be investigated, providing insight into the tradeoffs involved in fine tuning of the controller for different scenarios. Finally, the modularity and flexibility of the controller design is demonstrated by adding an obstacle avoidance controller, and testing its operation in conjunction with the existing manipulability and stability controller.

5.1 SIMULATION SETUP

The model parameters and controller gains used for all simulations in this chapter are the same ones presented in their respective chapters. A static load of 1.25 kg is applied at the end effector during all scenarios to simulate the mobile manipulator having grabbed an object. This load reduces the tip-over stability of the mobile manipulator, and thus allows the controller to be more thoroughly tested.

The randomly generated scenarios consists of 5 randomly generated waypoints in a confined space of 8 m by 8 m and 0.65 m high. The mobile manipulator is tasked with reaching these waypoints in a predetermined sequence. The height limit is to ensure that all generated waypoints are within physical reach of the manipulator. The size of the workspace area and number of waypoints per scenario was chosen to ensure each scenario has a good range of waypoint placements while also limiting the simulation duration required of each scenario so that many simulations can be performed in a timely manner. The waypoints in each scenario are randomly generated using the MATLAB random function, which produces random numbers in a uniform distribution. A set of 100 different simulation scenarios was created, each having 5 waypoints. The same set of simulation scenarios are used as the mobile manipulator thresholds are varied to see their effects on the performance of the mobile manipulator.

For each simulation, the base of the mobile manipulator starts at the global origin, pointed along the positive x -axis with the end effector in the default pose. The global origin is also aligned with the xy centre of the workspace area where the waypoints are generated. Each simulation has a time limit of one minute to ensure a simulation will end should the controller get stuck trying to reach a waypoint. The one minute simulation time limit was empirically obtained to be a sufficient time for a simulation to be completed for the given workspace area and number of waypoints. Figure 5.1 shows the top and side views of a simulation scenario made up of the five generated waypoints. The first waypoint is denoted by the diamond and the final waypoint denoted by the circle, and, as stated previously the mobile manipulator always starts at the origin. Figure 5.2 shows the total distance of the path defined by the waypoints in each of the simulation scenarios.

To conclude that the mobile manipulator was successful in each scenario, the following two conditions must be met:

1. All the waypoints must be reached in the given time limit to a tolerance of 0.01 m,
2. Absolute stability must be maintained above 1 for the entirety of the simulation.

Along with determining if the simulation was successful or not, the following data are recorded to evaluate and compare the operation of the mobile manipulator as the simulation parameters are varied:

- Simulation time,
- Simulation speed,
- Time spent moving base to target,
- Time spent moving manipulator to target,

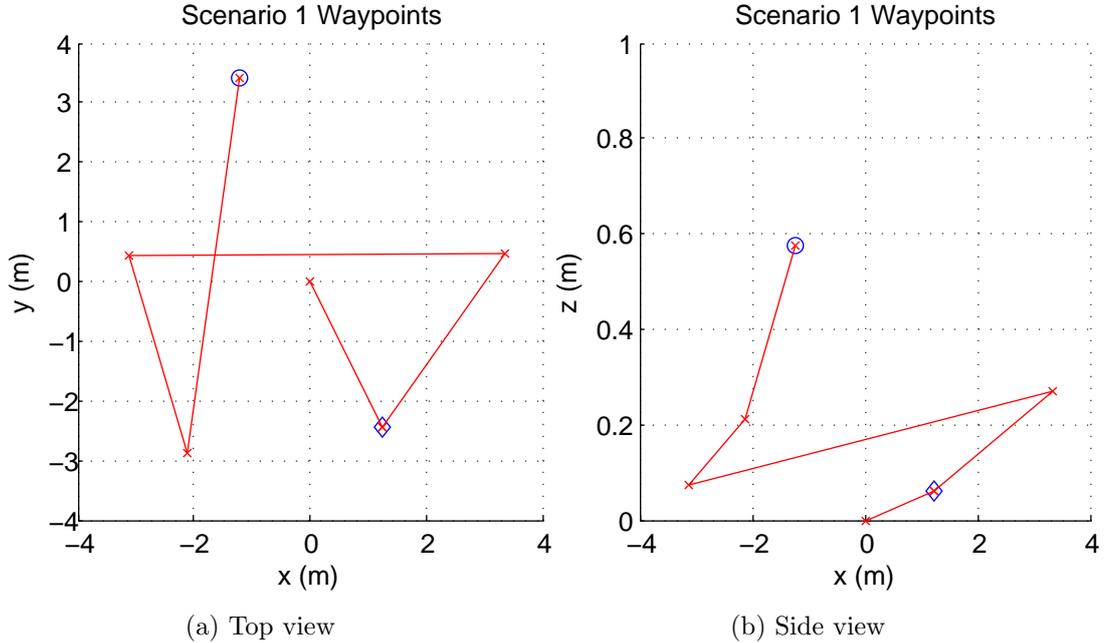


Figure 5.1: A set of generated waypoints for a simulation.

- Time spent in fixing manipulability state,
- Number of state transitions into fixing manipulability state,
- Time spent in fixing stability state,
- Number of state transitions into fixing stability state.

Simulation time is a straightforward measurement of how long it takes the mobile manipulator to reach all the waypoints in a simulation. It is used to compare the performance of the mobile manipulator under different simulation parameters for the same set of waypoints. Simulation speed is the straight-line distance from the starting location to all the waypoints, traversed in sequence, for the simulated scenario divided by the simulation time. This normalises the simulation times so that comparisons between different sets of waypoints can be made. The speed measurement can highlight the effect of different waypoint placements on the performance of the mobile manipulator; for example, if two simulations of different sets of waypoints have a similar simulation time, but the simulation speeds differ significantly, further analysis can then be done on these simulations to find out why. The measurement of time spent moving the base and arm to target describes the total time during the simulation the mobile manipulator is performing the actual task of reaching waypoints. These two times should not change drastically as they are dependent on the distance of each scenario and number of waypoints respectively, both of which are not altered between different simulations. The measurement of time spent in each state, that of manipulability and stability, provides a direct indication on the effects of altering the thresholds for those performance

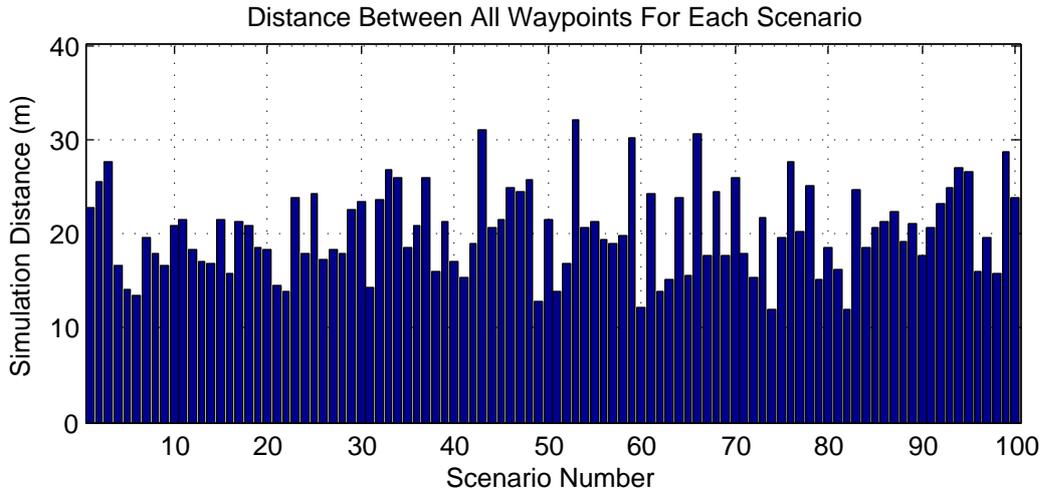


Figure 5.2: Distance of each set of generated waypoints used for the simulations.

metrics. The number of state transitions into the manipulability and stability states measurement is another different way to see the effects of altering the thresholds of the performance metrics, especially the hysteresis thresholds as they are designed to control the number of state transitions. All simulation data and results in the following sections are presented to an accuracy of 3 significant figures. Calculations on simulation data and results for comparison purposes are performed on actual simulation values and then rounded to 3 significant figures.

5.2 BASELINE SIMULATION

The first set of simulations establishes a baseline setup that later simulations can be compared against. The model parameters and controller gains used for this simulation are the same as presented in Chapter 3 (model) and Chapter 4 (controller) respectively. The baseline manipulability and stability thresholds, as well as the hysteresis thresholds used are listed in table 5.1 below.

Table 5.1: Baseline simulation: stability and manipulability thresholds

	Manipulability	Stability
Threshold	0.750	1.25
Hysteresis threshold	0.900	1.60

Table 5.2 shows the mean and standard deviation (in brackets) simulation results of the 100 scenarios performed. Where applicable, percentage values show the proportion of time spent in a particular state relative to the simulation time. Figures 5.3 to 5.9 shows the individual results of each simulation scenario. The controller was successfully able to complete all 100 scenarios under the 1 minute time limit as well as maintaining stability above 1 for the entire period during each scenario.

Table 5.2: Baseline simulation: mean and standard deviation results of all scenarios.

No. of failed scenarios	0
Simulation time (s)	44.9 (4.22)
Simulation speed (ms^{-1})	0.448 (0.0686)
Time spent moving base (s)	26.1 (58.1%) (4.24)
Time spent moving manipulator (s)	14.8 (33.0%) (0.981)
Time in manipulability state (s)	1.35 (3.01%) (0.903)
No. of manipulability state transitions	1.83 (1.09)
Time in stability state (s)	2.67 (5.94%) (1.12)
No. of stability state transitions	4.64 (1.53)

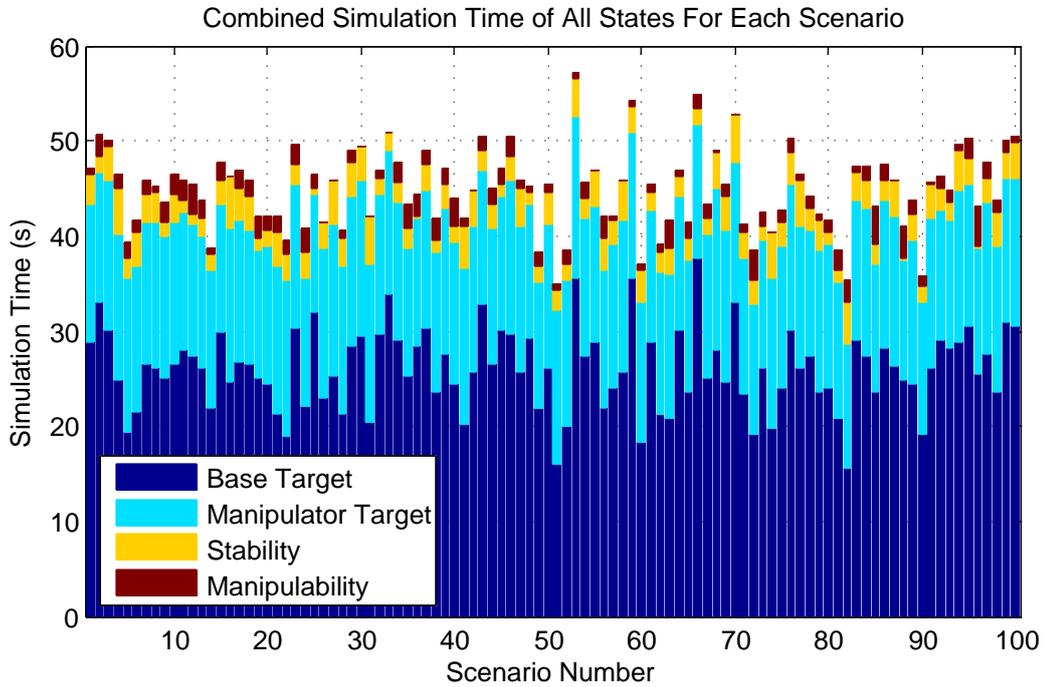


Figure 5.3: Baseline simulation: combined simulation time results of all states.

Figure 5.3 shows the time the mobile manipulator spent in each of the states. It shows the majority of the time is spent in driving the mobile base to the next waypoint. This time is affected by the total distance and the speed for the mobile base movement. The next largest amount of time spent is in moving the arm to the waypoint. Again, this is affected by the speed of the manipulator and the number of waypoints in the simulation. It is mostly from the moving arm to target state that transitions to the stability and manipulability states can occur, so a better representation of the effect of changing the manipulability and stability thresholds that will be used is to compare them with respect to the sum of the time spent in moving the manipulator to the waypoint, time spent in stability state and time spent in manipulability state. For the baseline simulation, the percentage of time spent in the manipulability and stability states compared to the sum of the times in the 3 states is 7.18% and 14.2% respectively.

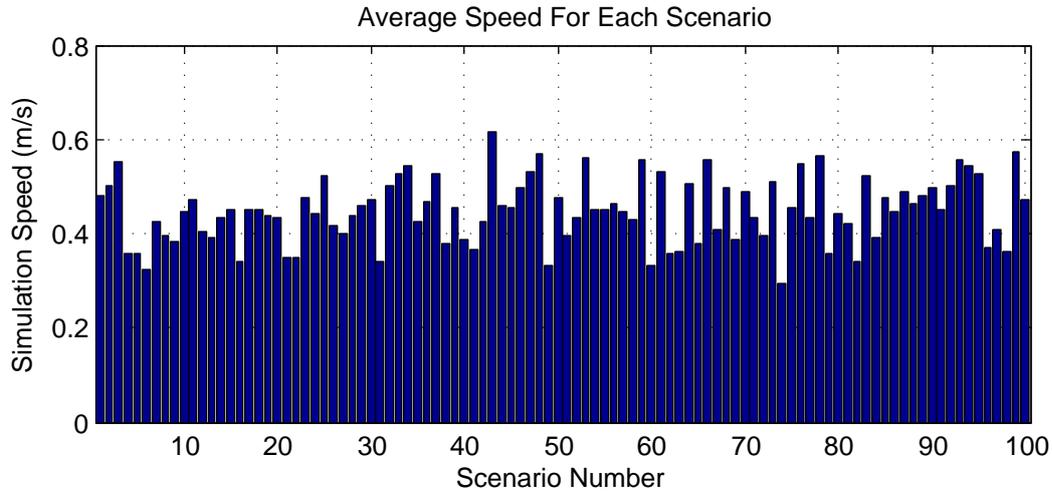


Figure 5.4: Baseline simulation: simulation speed results.

Figure 5.4 shows the simulation speed of the individual simulation scenarios, the variability of the speeds suggests that waypoint placement is another factor that can affect the overall performance of mobile manipulator. This is further shown in figure 5.5 when only the time spent in driving the base from waypoint to waypoint is used to calculate the mean speed, the variability is still present. This result is also evident in the standard deviation for the moving base state as it has a larger relative standard deviation compared to the moving arm state.

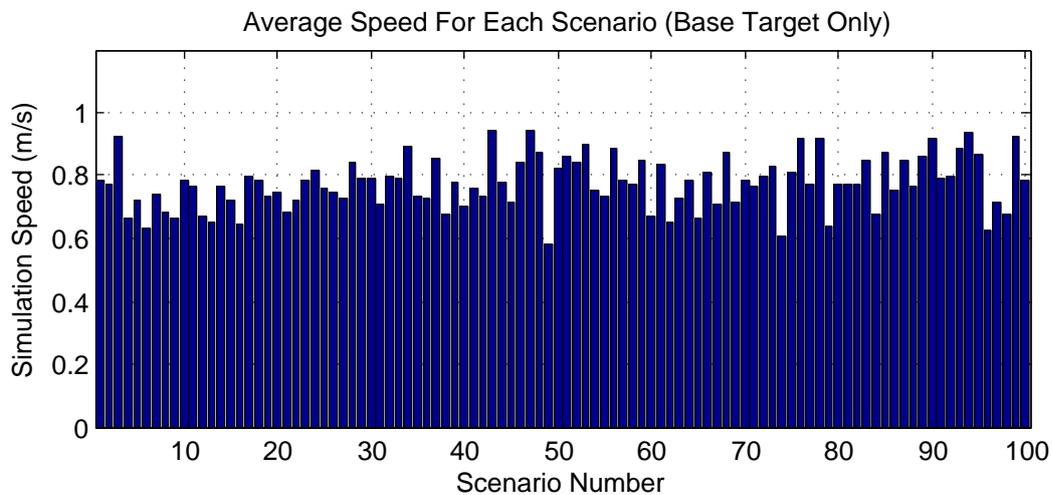


Figure 5.5: Baseline simulation: simulation speed using time spent in base moving to waypoint only.

Table 5.3: Baseline simulation: stability variations.

	Mean	Standard Deviation	% of Mean
Time in stability state	2.67 s	1.12 s	41.9%
Number of stability state transitions	4.64	1.53	33.1%

Stability results for this set of simulations in figure 5.7 show a range of results for the time spent in the stability states and stability state transitions. From the graphs, it looks like there is more variability in the stability time results compared with the stability state transitions. This is confirmed in table 5.3 which shows the standard deviation and the ratio to their respective means. This indicates that the time spent in stability transitions have more variability compared to stability state transitions, and that a large number of stability state transitions does not always result in a longer time spent in the stability state.

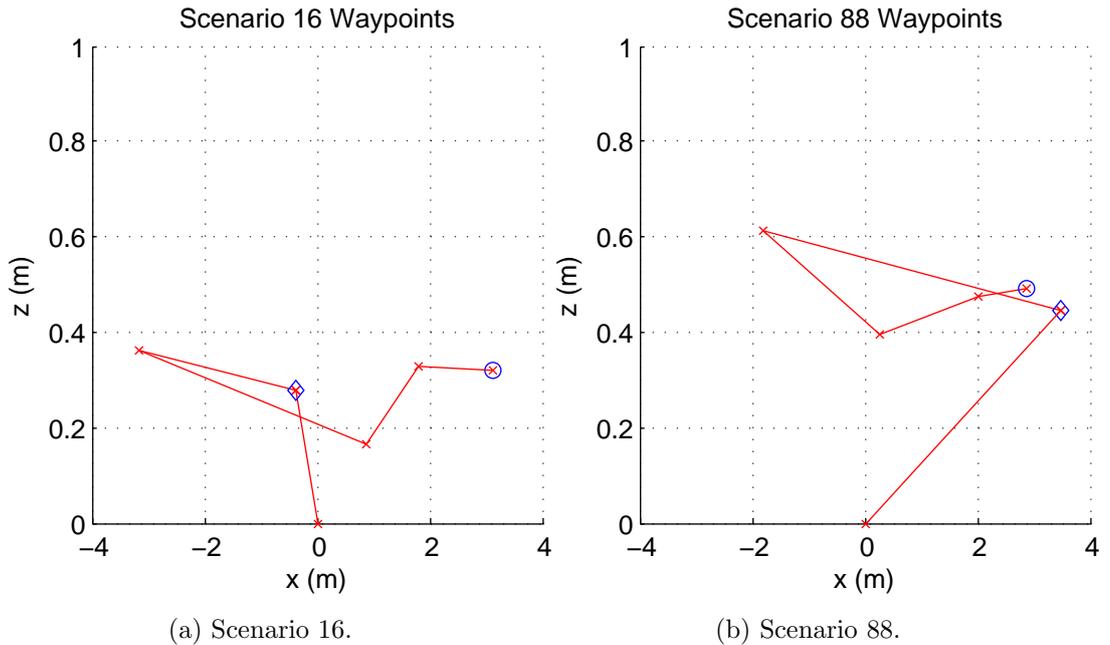


Figure 5.6: Comparison of waypoint height placements.

The individual scenarios that produced the shortest and longest time spent in the stability state are also investigated to see if there is a pattern in the waypoint placement that may have caused this. One pattern that has been identified is associated with the height of the waypoint placements. Waypoints placed higher in the workspace are less likely to cross the stability threshold than waypoints placed closer to the ground. An example of this is shown in figure 5.6 which shows the waypoint placements of scenario 16 and 88, where in scenario 16, 5.42 s was spent in the stability state compared with 0.200 s in scenario 88. The effect of waypoint height on stability performance is due to how stability is calculated and how the controller determines if a waypoint

is in range of the manipulator. The maximum range of the manipulator is fixed to be the sum of the length of the shoulder and elbow links, and a waypoint is in range if the straight-line distance between the waypoint to the base of the manipulator is less than the maximum range. This straight line distance will have both a vertical and horizontal component. The current stability metric is only dependent on the horizontal component, therefore, when a waypoint is further away horizontally, the manipulator will have to reach further out and increase the chance of crossing the stability threshold. A higher waypoint will have a smaller horizontal component relative to the mobile base even though the straight-line distance could still be the same, and so will reduce the chance of crossing the stability threshold.

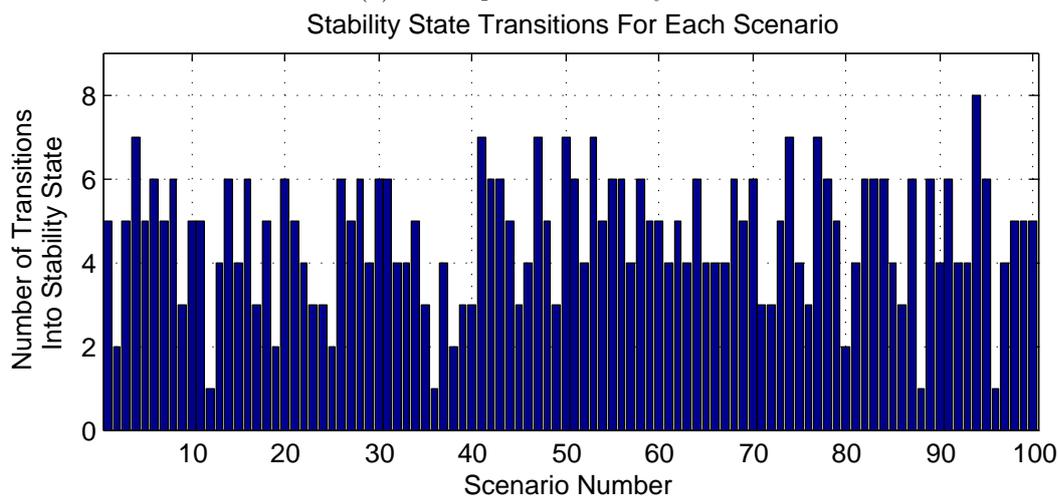
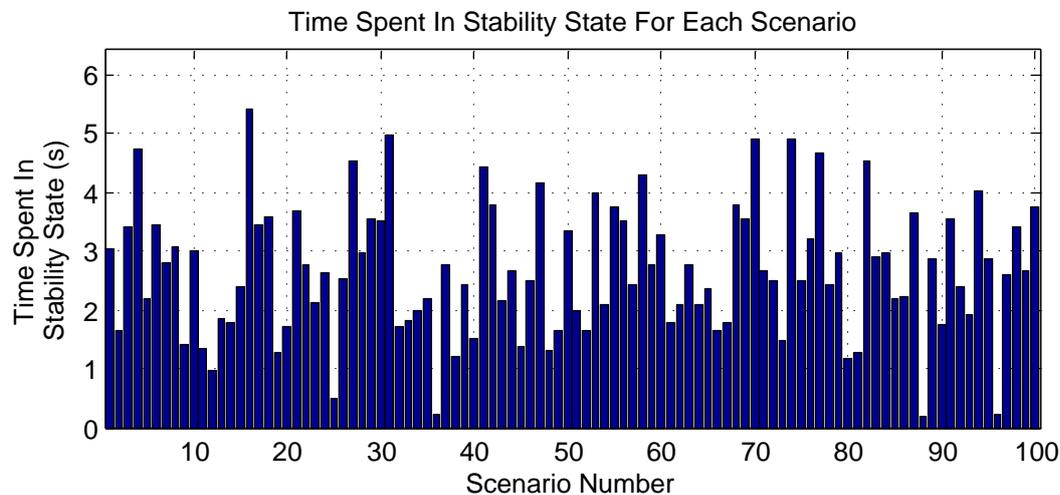


Figure 5.7: Baseline simulation: stability results.

Manipulability results are shown in figure 5.9. Again from the graphs, it looks like there is more variability in the time measurement compared to the state transitions,

with table 5.4 confirming this. The same conclusion that was made for the stability case is also applicable here. Waypoint placement was again investigated to identify any placement patterns that might explain what has caused high and low times spent in the manipulability state. It appears that the high times spent in the manipulability state correspond to the same scenarios where the time spent in the stability state is low, and vice versa. This is most likely because the priority system established for the controller prioritises stability over manipulability, therefore, for scenarios where the stability threshold is not often crossed, there are more opportunities for the manipulability controller to be executed. Another contributing factor is that the controller actions for improving manipulability and stability overlap, as a result, improving one will also improve the other. This is the reason why low times in the manipulability state and low manipulability state transitions occur for scenarios where the stability controller is triggered more often, to an extent where certain scenarios can have no executions of the manipulability controller or even cross the manipulability threshold. An example of this is shown in figure 5.8 for scenario 16 where the manipulability threshold is not crossed during the entire scenario.

Table 5.4: Baseline simulation: manipulability variations

	Mean	Standard Deviation	% of Mean
Time in manipulability state	1.35 s	0.903 s	66.8%
No. of manipulability state transitions	1.83	1.09	59.7%

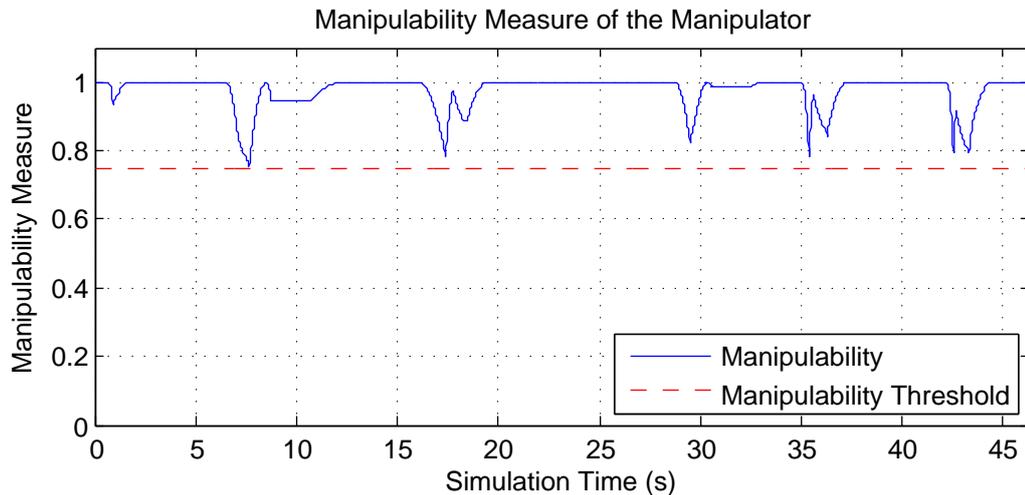
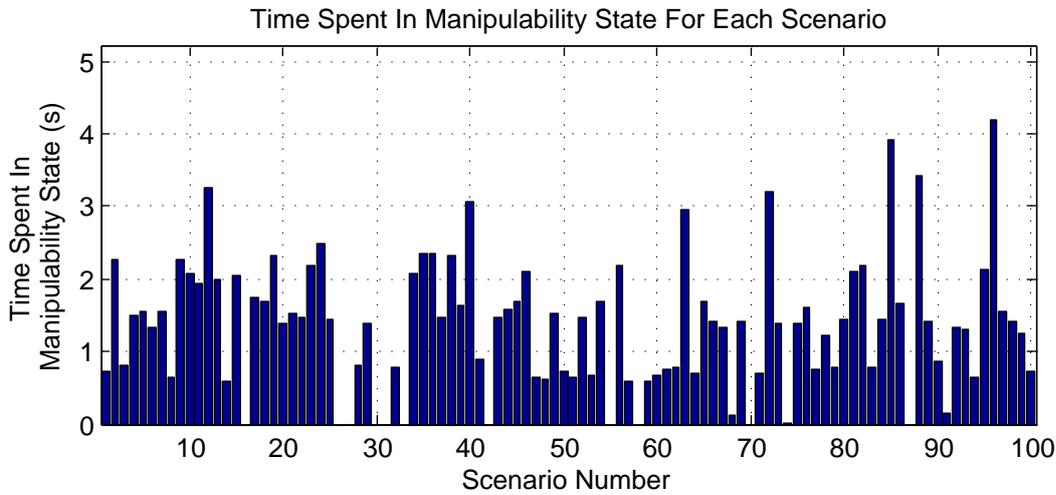
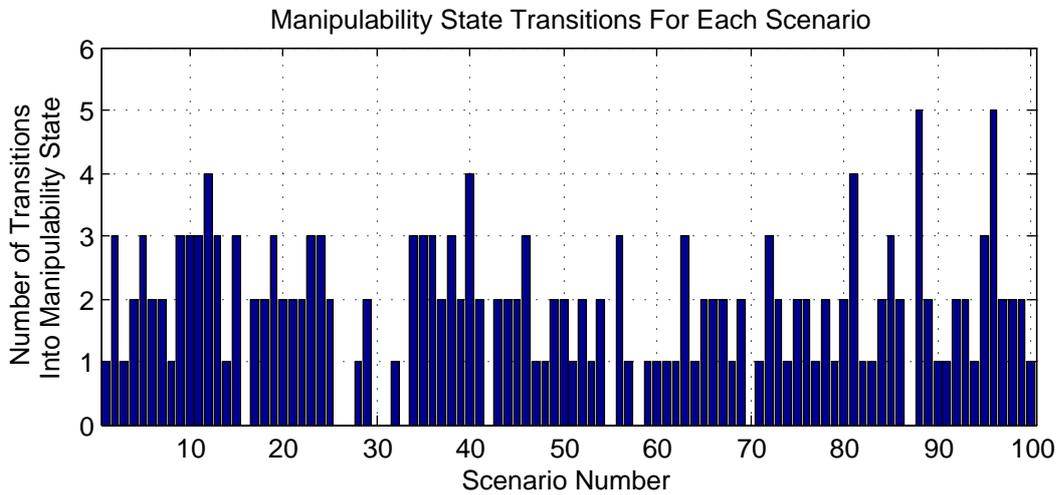


Figure 5.8: Baseline simulation: manipulability measure for scenario 16.



(a) Time spent in manipulability state.



(b) Number of transitions into manipulability state

Figure 5.9: Baseline simulation: manipulability results.

5.3 STABILITY THRESHOLD EFFECTS

The effects of changing the stability threshold on the performance of the controller is investigated in this section. The goal of these simulations is to further establish confidence in the controller for different threshold values, as well as to see what the tradeoffs are when selecting particular threshold values on performance. The following effects are investigated:

- Low stability thresholds,
- High stability thresholds,
- No stability hysteresis,

and the following sections present the results of these simulations.

5.3.1 Low and High Stability Thresholds

Low and high stability thresholds and their effects on controller performance are investigated first. Table 5.5 contains the new stability threshold values for the high and low stability simulations. The new stability thresholds are then used to rerun the simulation on the same set of 100 scenarios done previously. A summary of the results for these two simulations is presented in table 5.6 and figures 5.10 and 5.11.

Table 5.5: Low and high stability simulation: stability and manipulability thresholds.

	Manipulability	Low Stability	High Stability
Threshold	0.750	1.10	1.40
Hysteresis Threshold	0.900	1.30	1.70

Table 5.6: Stability simulations: mean and standard deviation results of all scenarios for low and high stability thresholds.

	Low stability	High stability
No. of failed scenarios	0	0
Simulation time (s)	41.9 (4.27)	46.6 (4.31)
Simulation speed (ms^{-1})	0.480 (0.0696)	0.431 (0.0647)
Time spent moving base (s)	26.3 (62.8%) (4.21)	25.7 (55.1%) (4.23)
Time spent moving manipulator (s)	13.3 (31.7%) (0.790)	15.7 (33.7%) (1.17)
Time in manipulability state (s)	1.97 (4.70%) (0.904)	0.637 (1.37%) (0.736)
No. of manipulability state transitions	2.86 (1.38)	0.800 (0.899)
Time in stability state (s)	0.318 (0.759%) (0.38)	4.60 (9.86%) (1.17)
No. of stability state transitions	1.53 (1.27)	8.57 (1.95)

In both simulations, the controller was again successful in completing every scenario without any breaches in stability or simulation time limits. A comparison of the results to the baseline simulation is shown in table 5.7. As expected, the major changes are to the time spent in stability states and the number of stability transitions. Figures 5.12 and 5.13 show the overall stability results of each scenario. The contrasting results of both simulations is evident with time spent in stability and stability state transitions much lower in the low stability simulation and much higher in the high stability simulation.

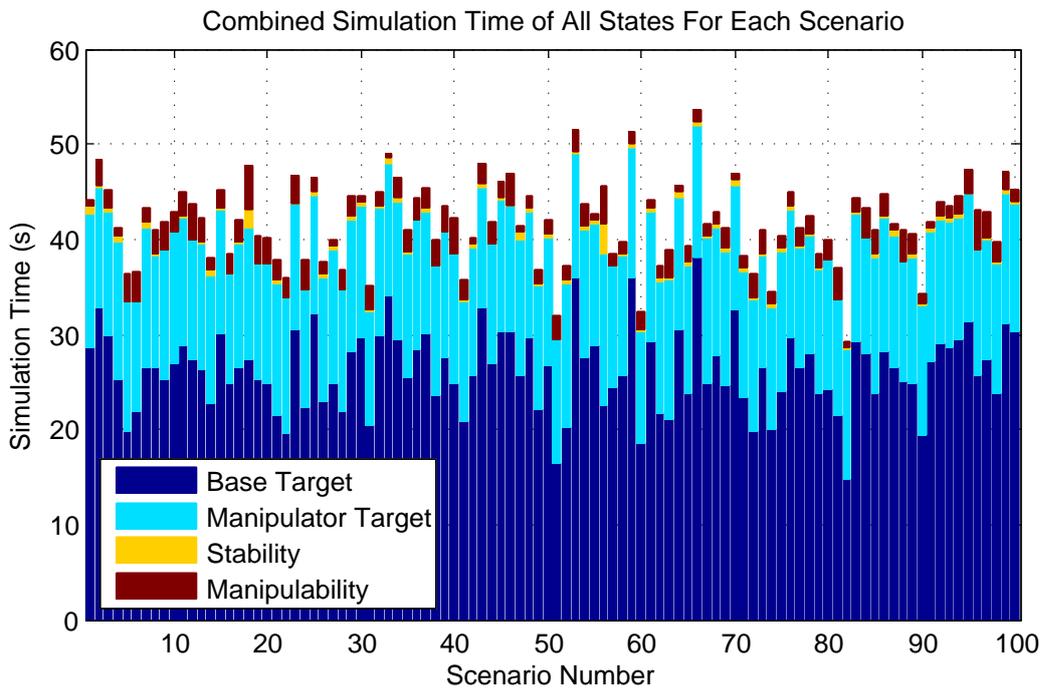


Figure 5.10: Low stability simulation: combined simulation time results of all states.

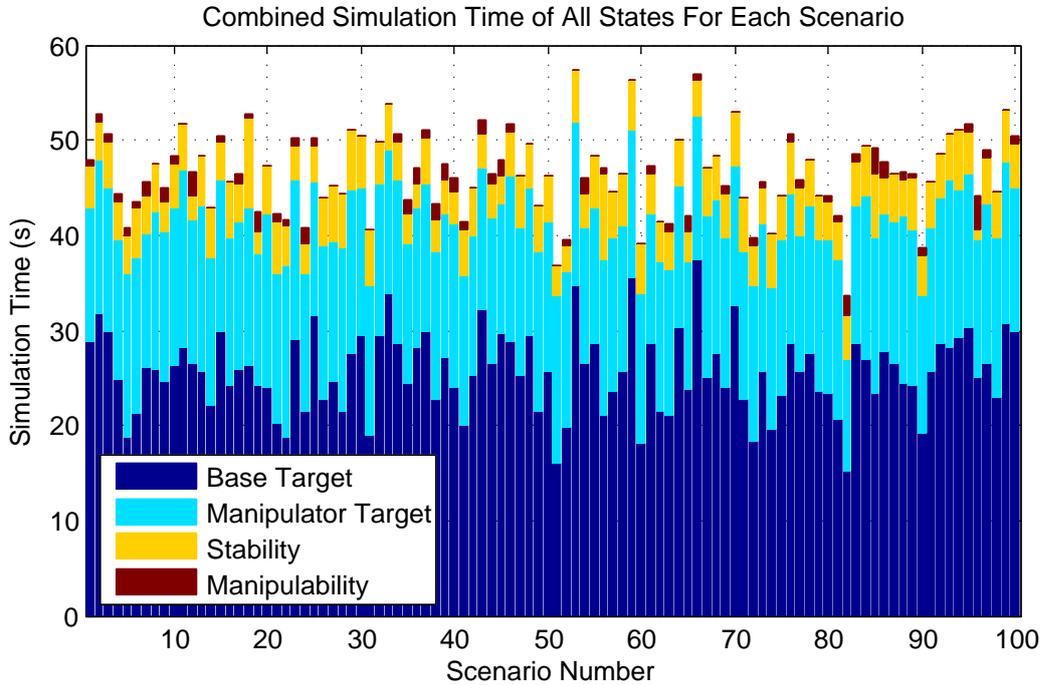


Figure 5.11: High stability simulation: combined simulation time results of all states.

Table 5.7: Comparison of high and low stability mean simulation results with the mean baseline simulation results.

	Low stability	High stability
Simulation time (s)	-3.00 (-6.68%)	+1.76 (+3.91%)
Simulation speed (ms^{-1})	+0.0318 (+7.10%)	-0.0172 (-3.83%)
Time spent moving base (s)	+0.237 (+0.908%)	-0.376 (-1.44%)
Time spent moving manipulator (s)	-1.50 (-10.1%)	+0.914 (+6.18%)
Time in manipulability state (s)	+0.617 (+45.7%)	-0.714 (-52.8%)
No. of manipulability state transitions	+1.03 (+56.3%)	-1.03 (-56.3%)
Time in stability state (s)	-2.35 (-88.1%)	+1.93 (+72.5%)
No. of stability state transitions	-3.11 (-67.0%)	+3.93 (+84.7%)

Comparing to the results of the baseline simulation for both the low and high stability simulations, the overall simulation time does not change much, with the low stability simulation taking 3.00 s (6.68%) on average shorter per scenario and the high stability simulation taking 1.76 s (3.91%) on average longer per scenario. There are no significant changes to the time and standard deviation results for driving the mobile base which is due to stability only having a minimal effect as the manipulator is kept stationary in this state. The time and standard deviation results for moving the arm is affected more, as it is mostly from this state that state transitions to the stability and manipulability states occur. Finally, there are bigger changes in the time and standard deviation results for the stability state which was the expected result of changing stability thresholds. Another more interesting change is the reduction of time spent in the manipulability state and number of manipulability state transitions for the high stability simulation and vice versa. It has been discussed previously that the controller actions for improving stability and manipulability overlap, and this effect can be seen here as while attempting to maintain a higher stability, the manipulability is improved at the same time. Figures 5.14 and 5.15 shows the manipulability results for the low and high stability simulations.

There are two scenarios which have caused outlier results for the stability measurements across both simulations. These are scenarios 18 and 56. Both scenarios in both simulations have much greater time spent in stability states compared to their respective simulation mean. Both scenarios were inspected to see what might cause the outlier result and, found that in both scenarios the controller is stuck on a pair of waypoints that are placed very close to each other but require a large heading change to reach. During this period the controller oscillates between the target, manipulability and stability states until it eventually works its way out. The problematic waypoints are 1 and 2 in scenario 18 and 3 and 4 in scenario 56, as shown in figure 5.16. An example of the state transitions that is occurring during scenario 56 is shown in figure 5.17. The oscillations described start at about 28 s into the scenario and end at about 35 s.

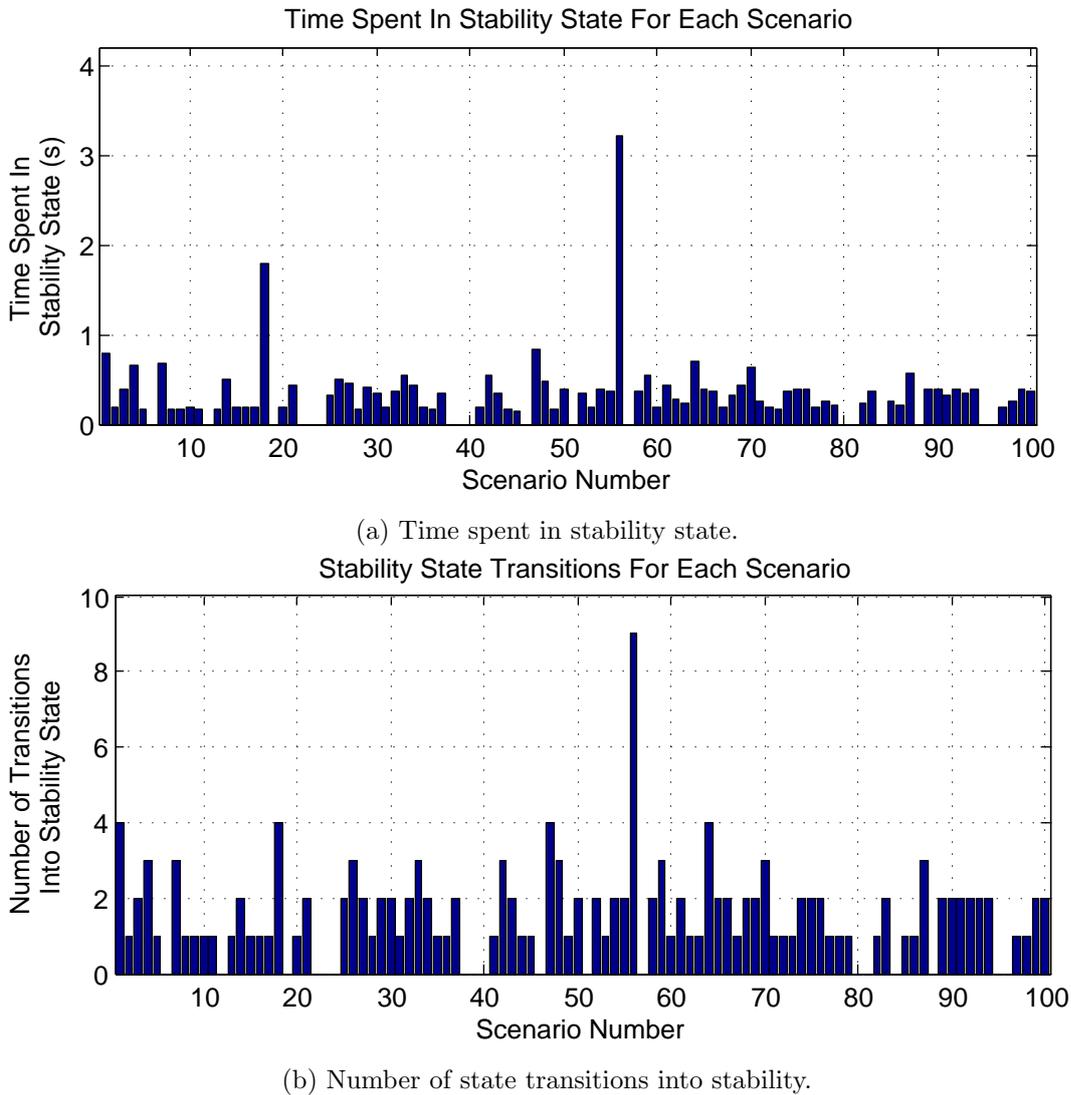


Figure 5.12: Low stability simulation: stability results.

As mentioned previously, a better representation of the effect of changing thresholds values is to compare only to the time spent moving the manipulator to the target, and not the overall simulation time. This comparison is shown in table 5.8 and the effects of different threshold values are more prominent. The controller is in the stability state 2.03% and 21.9% of the time for the low and high stability simulations respectively. When the controller is in the stability state, it is not executing the current task, which introduces overheads and inefficiencies in terms of performance. The tradeoff from the decreased performance is the reduced risk of tip-over.

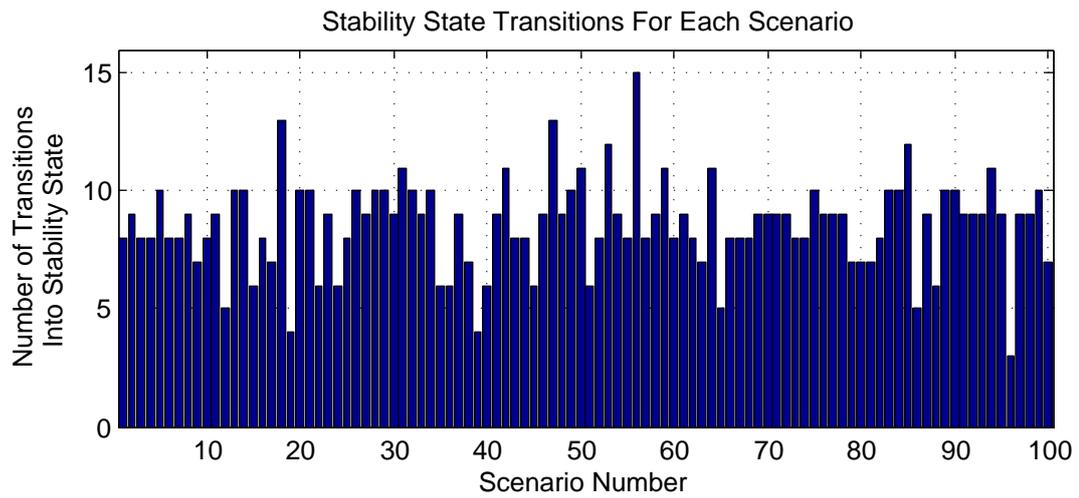
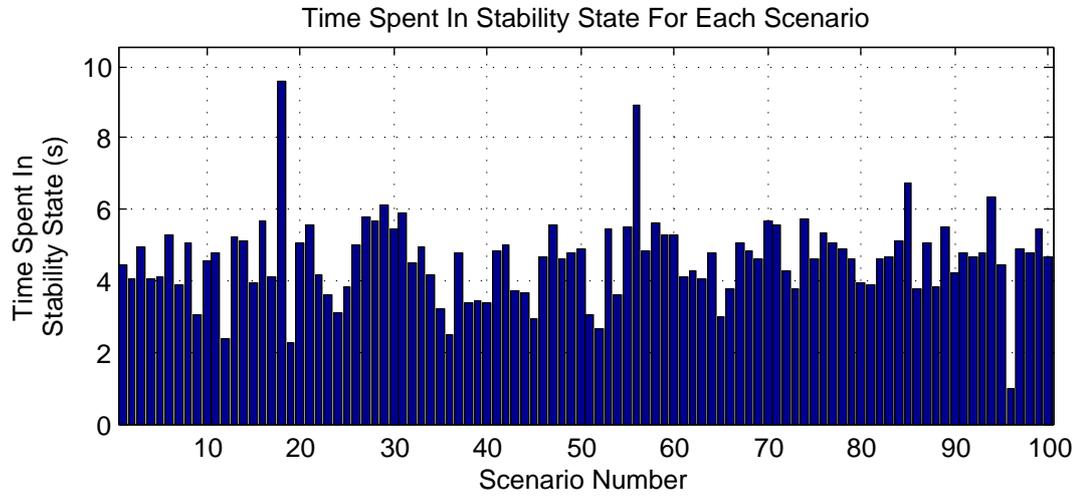


Figure 5.13: High stability simulation: stability results.

Table 5.8: Comparison of mean time spent in manipulability and stability states to total time spent moving manipulator to target, time in manipulability and time in stability only.

	Baseline	Low Stability	High Stability
Time in manipulability state	7.18%	12.6%	3.04%
Time in stability state	14.2%	2.03%	21.9%
Time spent moving manipulator	78.7%	85.3%	75.0%
Total time (s)	18.8	15.6 (-17.2%)	21.0 (+11.3%)

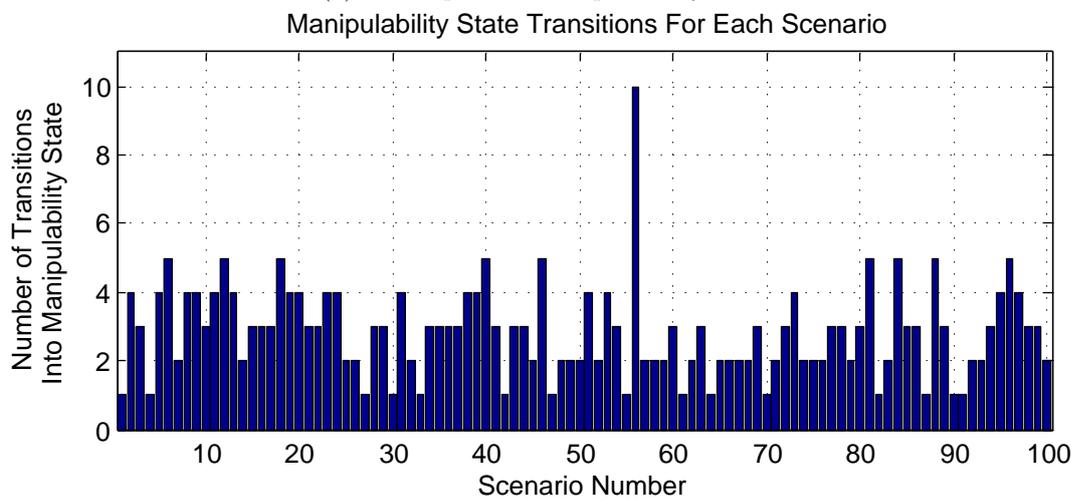
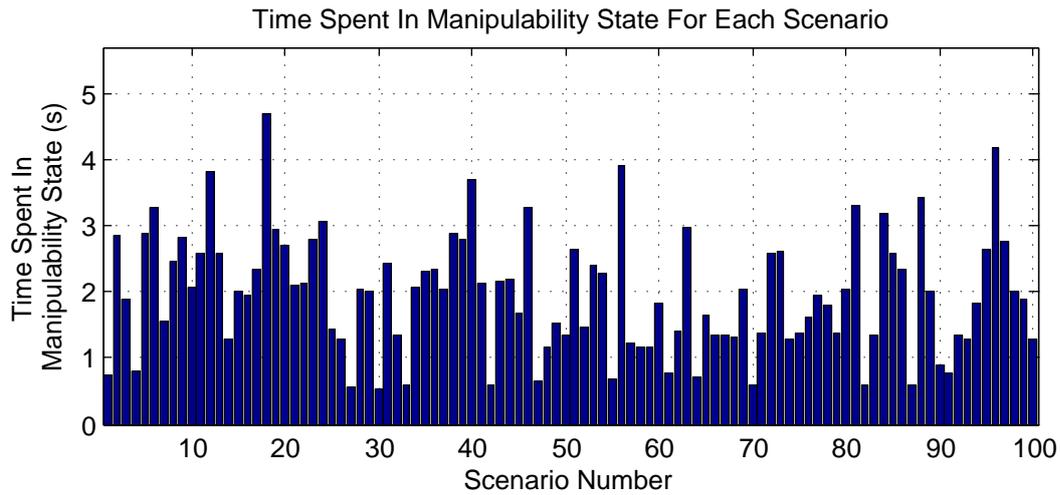
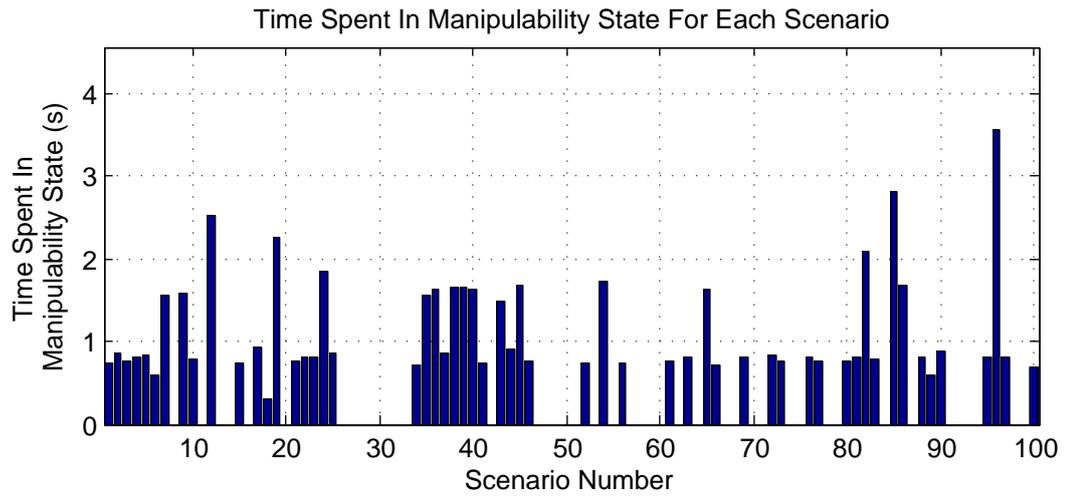
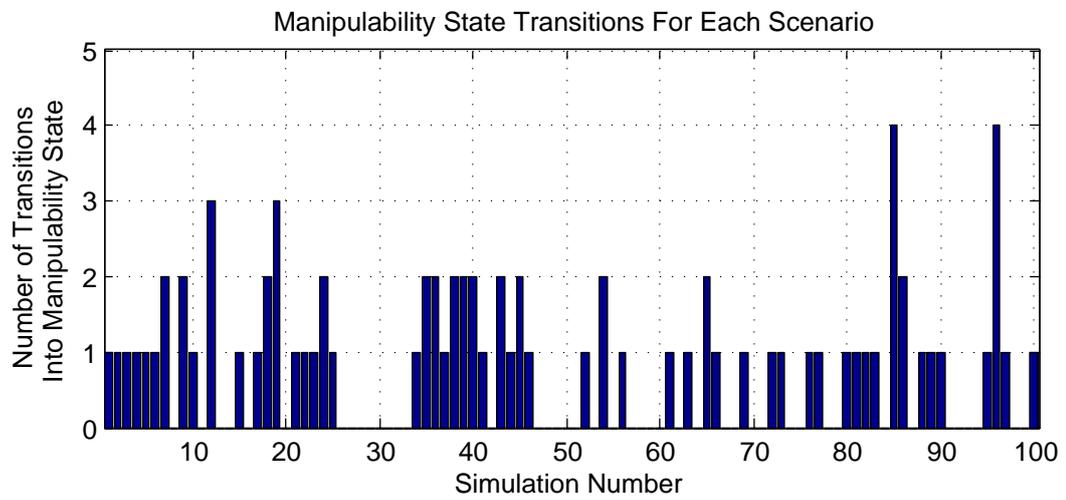


Figure 5.14: Low stability simulation: manipulability results.



(a) Time spent in manipulability state.



(b) Number of state transitions into manipulability.

Figure 5.15: High stability simulation: manipulability results.

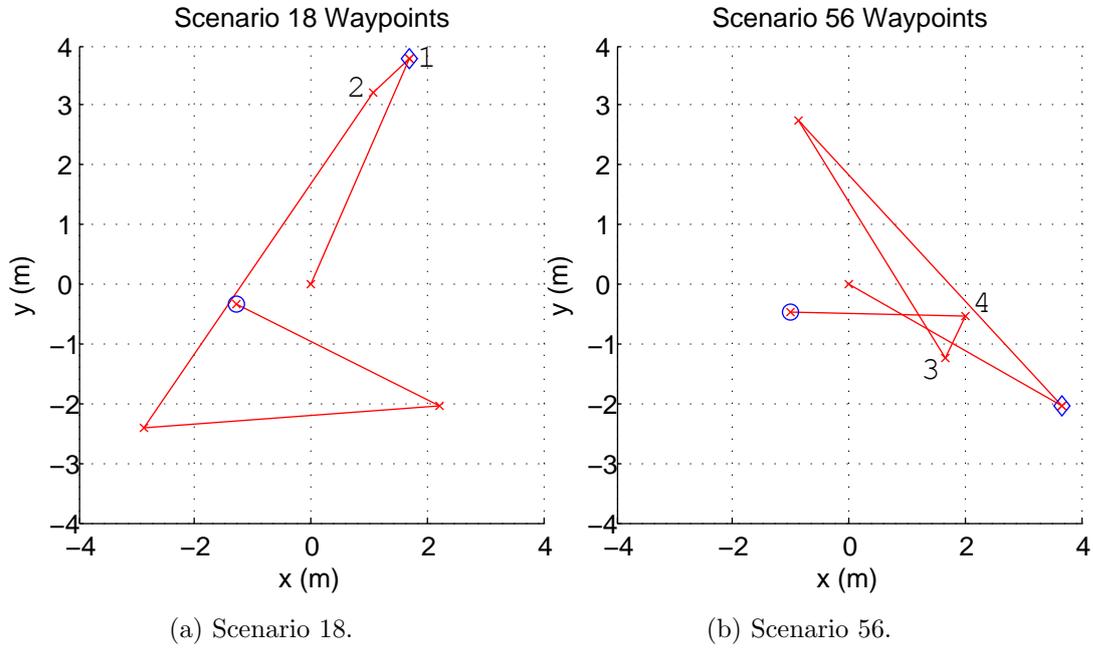


Figure 5.16: Scenarios causing outlier stability results for low and high stability simulation.

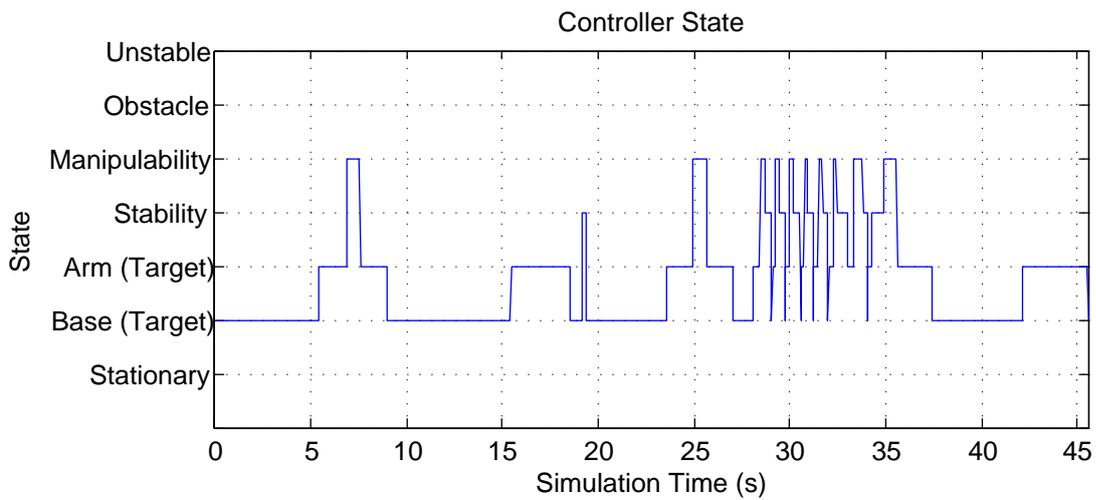


Figure 5.17: Low stability simulation: controller state for scenario 56.

5.3.2 No Stability Hysteresis

The aim of the hysteresis threshold is to reduce the number of state transitions into a particular state. The effect of removing the stability hysteresis and how effective it has been in reducing the state transitions is investigated in this section. Table 5.9 shows the thresholds used for this simulation.

Table 5.9: No stability hysteresis simulation: stability and manipulability thresholds.

	Manipulability	Stability
Threshold	0.750	1.25
Hysteresis threshold	0.900	None

Results for the simulation is shown in table 5.10 and figure 5.18. The two significant results are the large increase in the number of stability state transitions and corresponding standard deviation, and the reduced time spent in the stability state. The increase in state transitions is what was expected, and is what the hysteresis is designed to reduce, but it is evident that this also comes with a time performance penalty with the time spent in the stability state falling by 42.3% when hysteresis is removed. Another side effect of removing hysteresis for the stability threshold is the increase in the manipulability state transitions. This is due to the greater possibility of transitioning from the stability to manipulability state, as the controller is only briefly in the stability state before it has reached the target stability threshold, whereby the manipulability controller can then be executed if required. With hysteresis, by the time the stability is improved to the higher threshold, the manipulability has also gone above its threshold. A comparison of the controller state with and without stability hysteresis during a single scenario is shown in figure 5.19.

Table 5.10: No hysteresis stability simulation: mean and standard deviation results of all scenarios with no hysteresis for stability.

	Results	Comparison with baseline (mean)
No. of failed scenarios	0	0
Simulation time (s)	43.6 (4.08)	-1.26 (-2.81%)
Simulation speed (ms^{-1})	0.461 (0.0700)	+0.0129 (+2.87%)
Time spent moving base (s)	26.3 (60.2%) (4.21)	+0.189 (+0.717%)
Time spent moving manipulator (s)	14.4 (33.0%) (0.833)	-0.394 (-2.66%)
Time in manipulability state (s)	1.42 (3.26%) (0.945)	+0.0728 (+5.38%)
No. of manipulability state transitions	2.48 (2.03)	+0.65 (+35.5%)
Time in stability state (s)	1.54 (3.53%) (0.787)	-1.13 (-42.3%)
No. of stability state transitions	26.9 (11.3)	+22.3 (+480%)

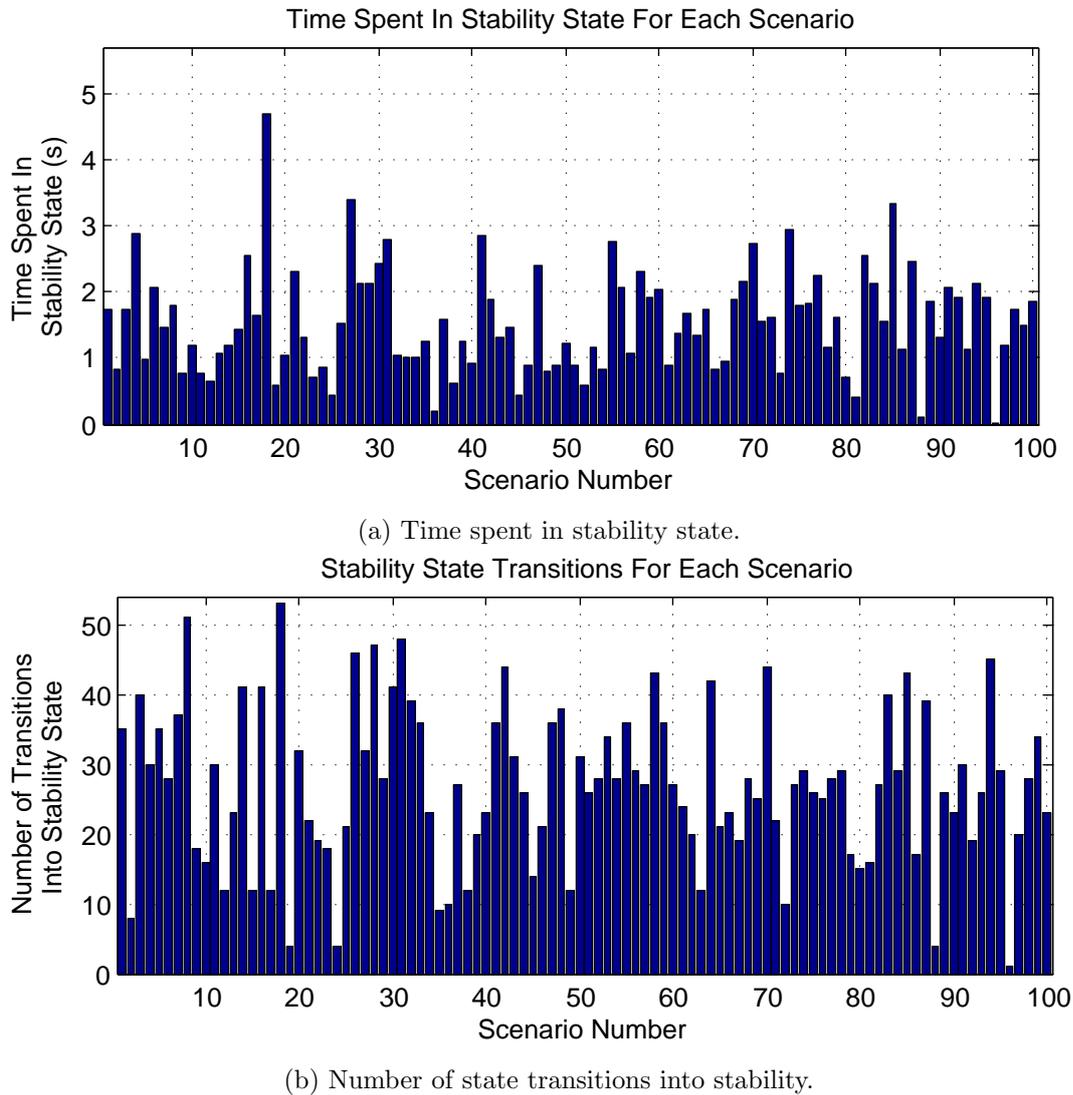
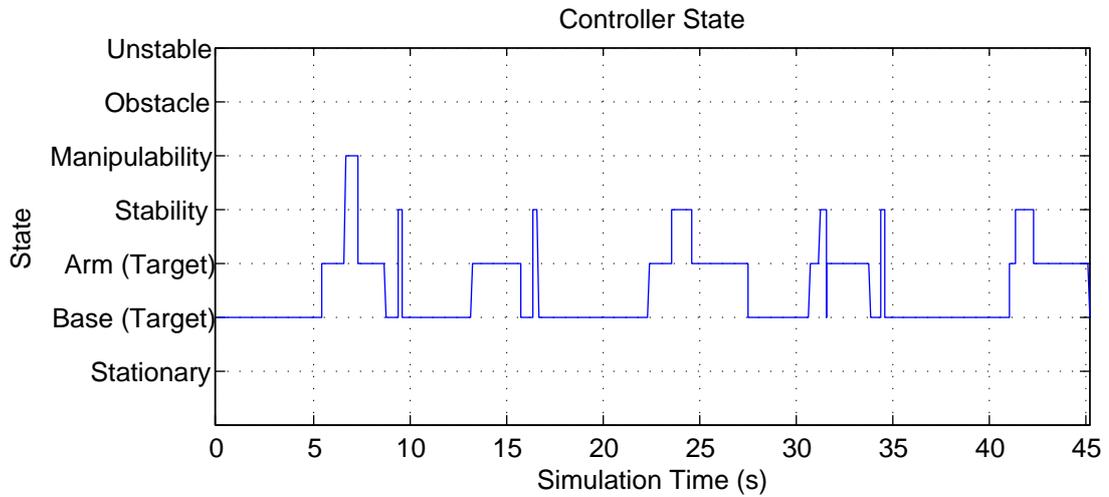


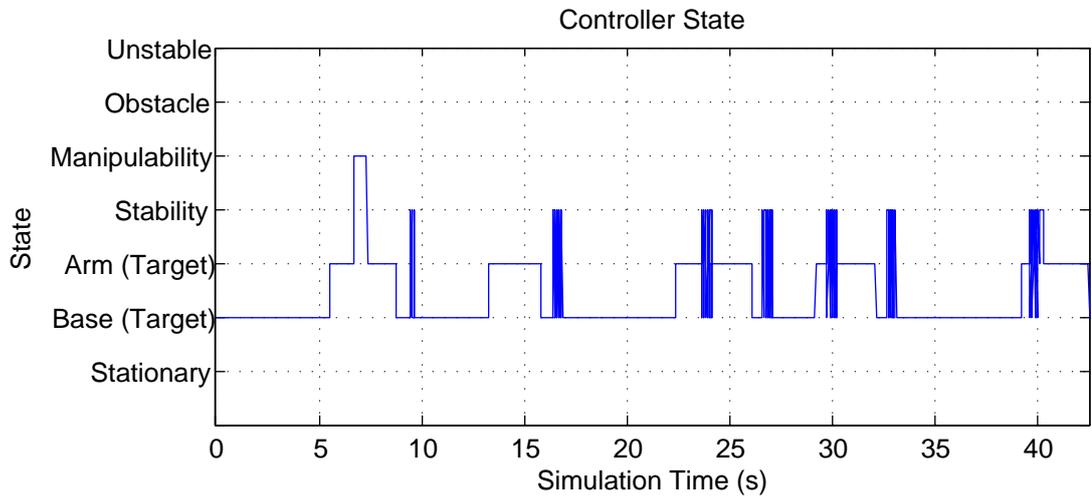
Figure 5.18: No stability hysteresis simulation: stability results.

The results have shown that state transitions increase drastically with no hysteresis, but the controller gain time performance. The magnitude of the physical effect of lots of controller state transitions is dependant on the actual hardware, but a common effect is inducing vibrations in the system. In terms of controller effects, oscillations can cause instability or long controller loops. A range of stability threshold values with no hysteresis as shown in table 5.11 is tested to see their effects on the controller.

From the results shown in table 5.12, thresholds 1 to 3 are fine with no stability hysteresis, but some scenarios start to fail using threshold 4 and many fail when using threshold 5. All of these failures come under the “failed to complete the simulation in time” category. Some of these scenarios exhibited an infinite loop changing between stability, manipulability, and target states. In other scenarios there was a long loop, but the controller was eventually able to break out to continue to the next waypoint,



(a) Baseline simulation: controller state for scenario 8.



(b) No stability hysteresis simulation: controller state for scenario 8.

Figure 5.19: Comparison of controller state with and without stability hysteresis for scenario 8.

however, this meant there was not enough time to complete the rest of the scenario. When the high stability simulation was performed, stability thresholds of 1.40 and 1.70 for hysteresis were used, with the result being no failed scenarios. The same is done with threshold 5 where stability thresholds of 1.50 and 1.70 for hysteresis were used and this time there were no failed simulations. A conclusion that can be made from this simulation result is that if one wishes to have high stability thresholds, then hysteresis should be used to ensure the scenario can be completed.

Table 5.11: No stability hysteresis simulation: range of stability and manipulability thresholds.

	Stability	Manipulability
Threshold 1	1.10	0.750
Threshold 2	1.20	
Threshold 3	1.30	
Threshold 4	1.40	
Threshold 5	1.50	
Hysteresis Threshold	None	0.900

Table 5.12: No stability hysteresis simulation: number of failed scenarios for each simulation.

	No. of failed scenarios
Threshold 1	0
Threshold 2	0
Threshold 3	0
Threshold 4	6
Threshold 5	42

5.4 MANIPULABILITY THRESHOLD EFFECTS

The effects of manipulability thresholds on the controller is investigated. The following sections present the results for:

- Low manipulability thresholds,
- High manipulability thresholds,
- No manipulability hysteresis.

5.4.1 Low and High Manipulability Thresholds

Again, the manipulability thresholds are first varied and the value used for the 100 scenarios are shown in table 5.13.

Table 5.13: Low and high manipulability simulation: stability and manipulability thresholds.

	Low Manipulability	High Manipulability	Stability
Threshold	0.600	0.900	1.25
Hysteresis threshold	0.700	0.950	1.60

A summary of the results of these simulations are presented in tables 5.14 and 5.15, and figures 5.20 and 5.21. The mean results for time spent and number of state transitions for manipulability followed the expected trend when compared to the baseline simulation, with less in the low manipulability simulation and more in the high

manipulability simulation and vice versa for the stability results. The standard deviation results for manipulability also followed the expected trend, but this was not the case for stability. The effect of the manipulability threshold on stability variability however, is minimal and suggests manipulability has a lesser effect on stability than the reverse situation. Figures 5.22 to 5.25 shows the manipulability and stability results for each scenario. A drastic, but expected result is highlighted in figure 5.22 with only 11 scenarios even triggering the manipulability controller due to the low manipulability threshold that was set.

Table 5.14: Manipulability simulations: mean and standard deviation results of all scenarios for low and high manipulability thresholds.

	Low manipulability	High manipulability
No. of failed scenarios	0	0
Simulation time (s)	44.9 (4.23)	44.1 (4.24)
Simulation speed (ms^{-1})	0.448 (0.0685)	0.456 (0.0677)
Time spent moving base (s)	25.7 (57.2%) (4.24)	26.1 (59.3%) (4.24)
Time spent moving manipulator (s)	16.2 (36.0%) (0.914)	13.3 (30.2%) (0.964)
Time in manipulability state (s)	0.0474 (0.106%) (0.153)	2.29 (5.20%) (1.01)
No. of manipulability state transitions	0.11 (0.314)	4.28 (1.33)
Time in stability state (s)	3.01 (6.70%) (1.04)	2.32 (5.26%) (1.09)
No. of stability state transitions	5.33 (1.33)	4.08 (1.51)

An unexpected result is the lower mean simulation time of each scenario in the high manipulability simulations compared with both the low manipulability and base-line simulations. The reduction in mean simulation time is due to the large fall in the time spent in the moving manipulator to target state. This is also present in the low stability simulation, which relates to the high manipulability simulation in the fact that the manipulability controller was executed more frequently and for longer periods for both cases. It is speculated that the link between a simulation with high manipulability controller execution and reduced time spent in moving manipulator to target is due to soft and hard threshold limitations on the manipulability and stability controllers respectively. The soft threshold used for the manipulability controller allows the manipulability to drop below the set threshold. This means that the mobile manipulator can still execute the current task, and, in the case shown here, reach for the waypoint, while the manipulability controller is improving the manipulability. This is not the case for the stability controller, where it must stop executing the current task and execute the stability controller due to the hard threshold limitation set. In both cases, any further task execution once the threshold is reached and even with their respective controller activated, can further reduce the respective performance metrics. This is the reason a hard threshold is used for the stability controller as it is important for the stability metric to not drop below the stability threshold.

Table 5.15: Comparison of high and low manipulability mean simulation results with the mean baseline simulation results.

	Low manipulability	High manipulability
Simulation time	+0.0294s (0.0655%)	-0.818s (-1.82%)
Simulation speed (ms^{-1})	-0.000275 (-0.0614%)	+0.00812 (+1.81%)
Time spent moving base (s)	-0.369 (-1.42%)	+0.0686 (+0.263%)
Time spent moving manipulator (s)	+1.36 (+9.18%)	-1.48 (-10.0%)
Time in manipulability state (s)	-1.30 (-96.5%)	+0.942 (+69.7%)
No. of manipulability state transitions	-1.72 (-94.0%)	+2.45 (+134%)
Time in stability state (s)	+0.345 (+12.9%)	-0.347 (-13.0%)
No. of stability state transitions	+0.69 (+14.9%)	-0.56 (-12.1%)

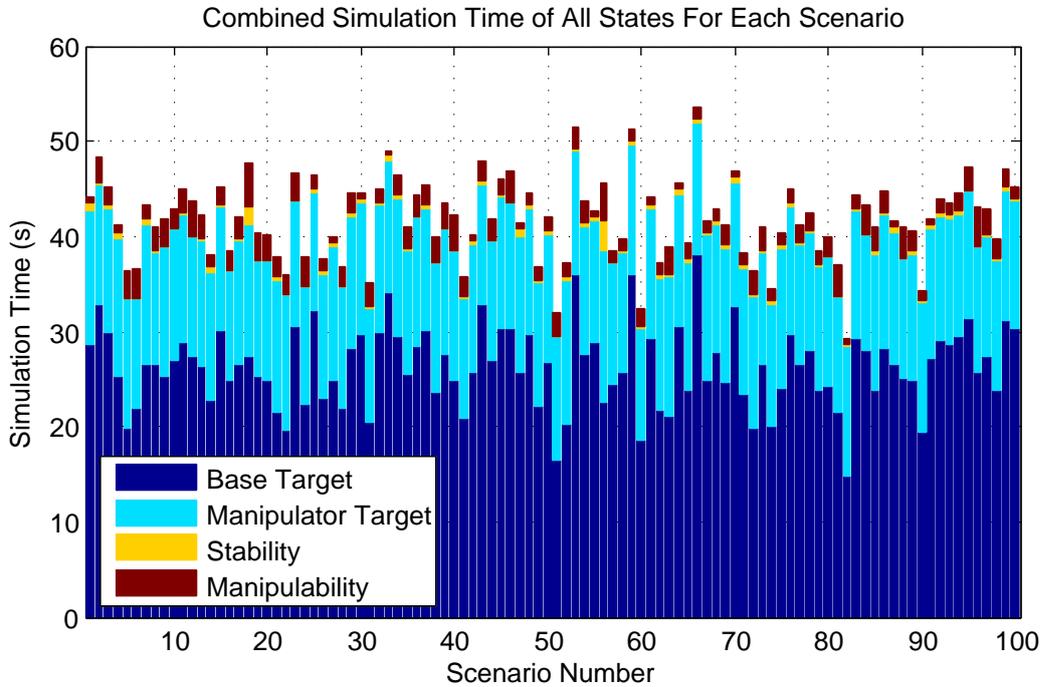


Figure 5.20: Low manipulability simulation: combined simulation time results of all states.

There are no obvious outliers in the individual scenario results shown in figures 5.22 to 5.25. Stability results of both simulations have a similar shape compared to the baseline simulation, but with different peak values. The low manipulability simulation only had 11 scenarios where the manipulability state was even entered into. The high manipulability simulation also had a similar shape compared to the baseline simulation, but with higher peak values. The similarity of the shapes that the individual results create is expected as a major influence on the individual results is of the waypoints placements themselves, which are kept the same through all the simulations. The absence of manipulability outliers suggests again that manipulability have a lesser effect on the controller compared to stability.

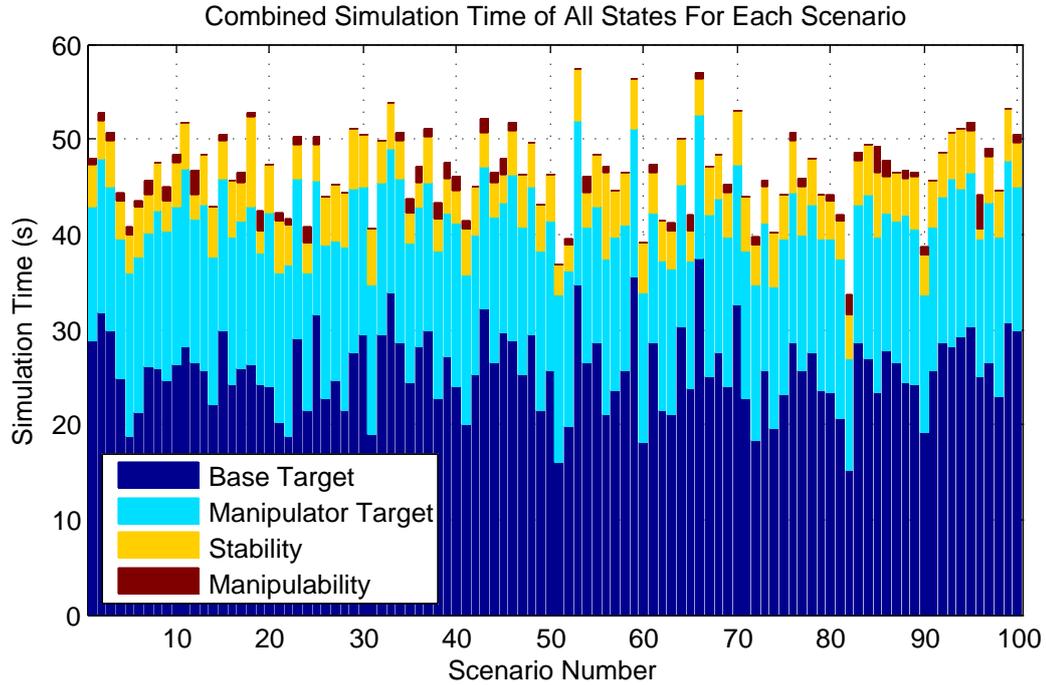


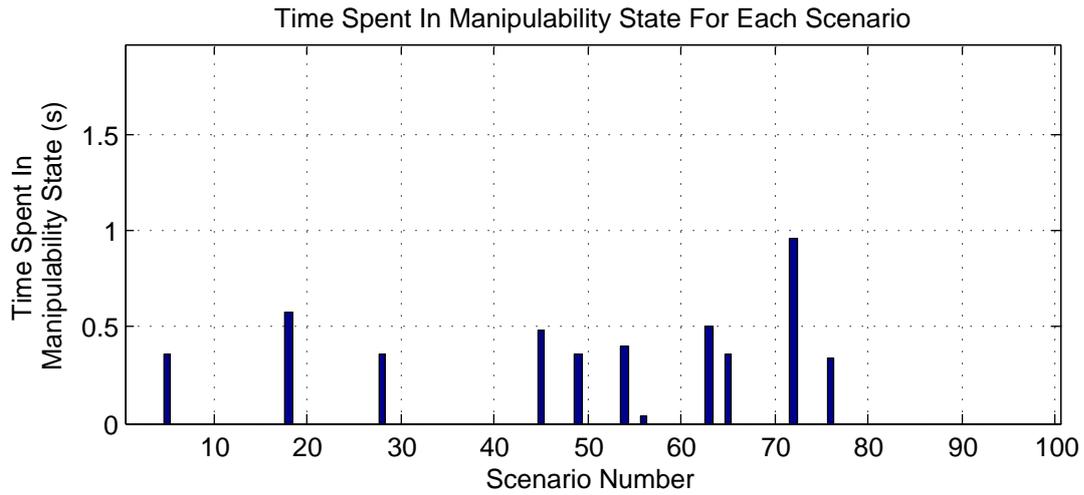
Figure 5.21: High manipulability simulation: combined simulation time results of all states.

Table 5.16: Comparison of mean time spent in manipulability and stability states to total time spent moving manipulator to target, time in manipulability and time in stability only.

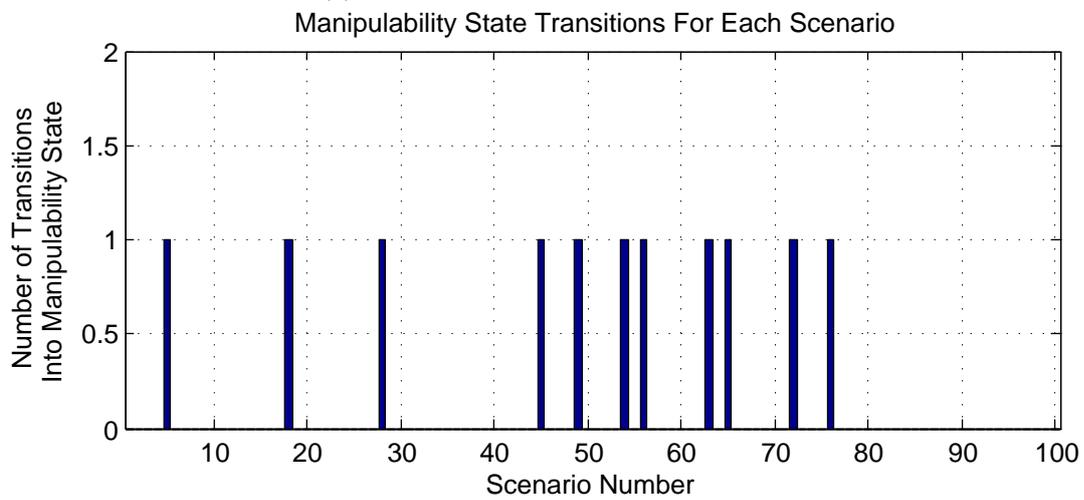
	Baseline	Low Manipulability	High Manipulability
Time in manipulability state	7.18%	0.247%	12.8%
Time in stability state	14.2%	15.7%	12.9%
Time spent moving manipulator	78.7%	84.1%	74.3%
Total time (s)	18.8	19.2 (+2.11%)	17.9 (-4.71%)

Table 5.16 shows the time comparison only including moving manipulator, manipulability and stability states. From the distribution of the times, it appears that the effect of low and high manipulability on stability with the baseline stability threshold is not as great when compared the other way around. This can be seen in the small change of $\pm 1.5\%$ in the time in stability state for the manipulability simulations, compared with a $\pm 5.4\%$ change in the time in manipulability state for the stability simulations. The change in total time compared with the baseline simulation is also much more for the stability simulations compared with the manipulability simulations, with the stability simulations causing greater than 10% changes compared to the less than 5% changes for the manipulability simulations. This indicates that, in the current setup of the model and the controller, stability has a higher effect on performance

than manipulability. Using this knowledge, one can specifically optimise parts of the controller that can have a greater effect on performance first.

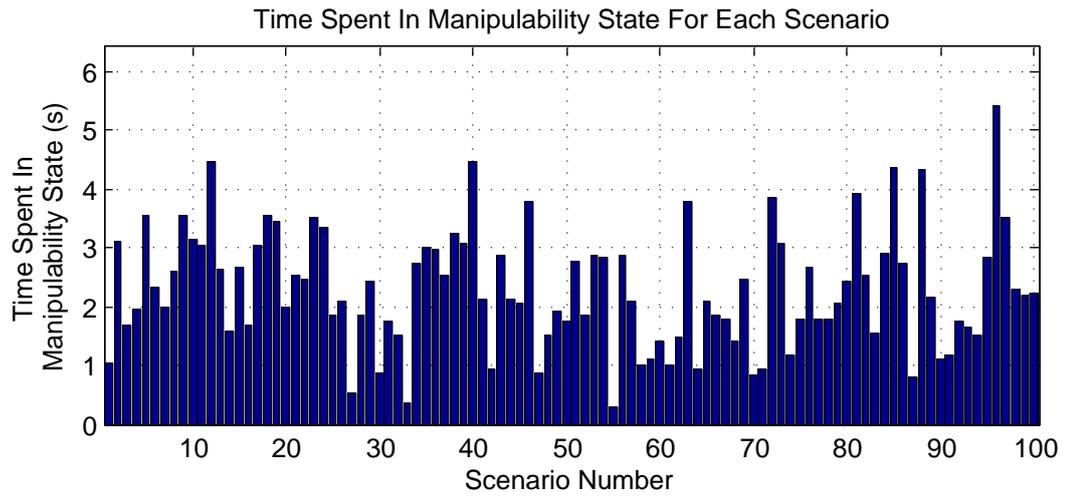


(a) Time spent in manipulability state.

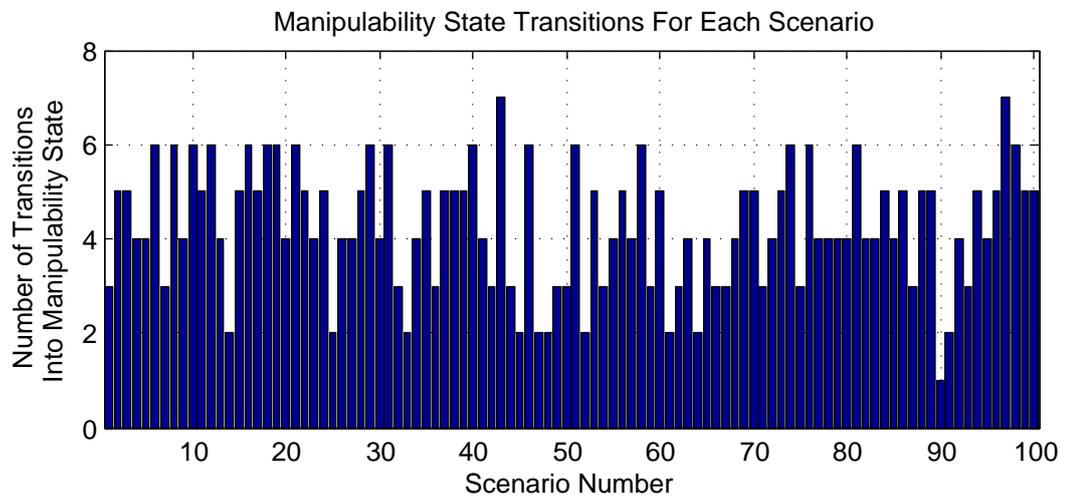


(b) Number of state transitions into manipulability.

Figure 5.22: Low manipulability simulation: manipulability results.



(a) Time spent in manipulability state.



(b) Number of state transitions into manipulability.

Figure 5.23: High manipulability simulation: manipulability results.

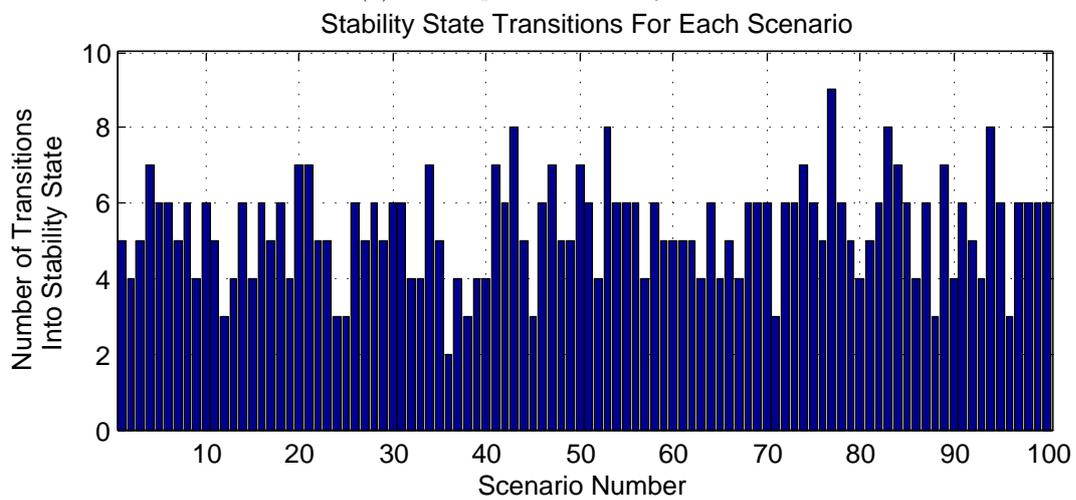
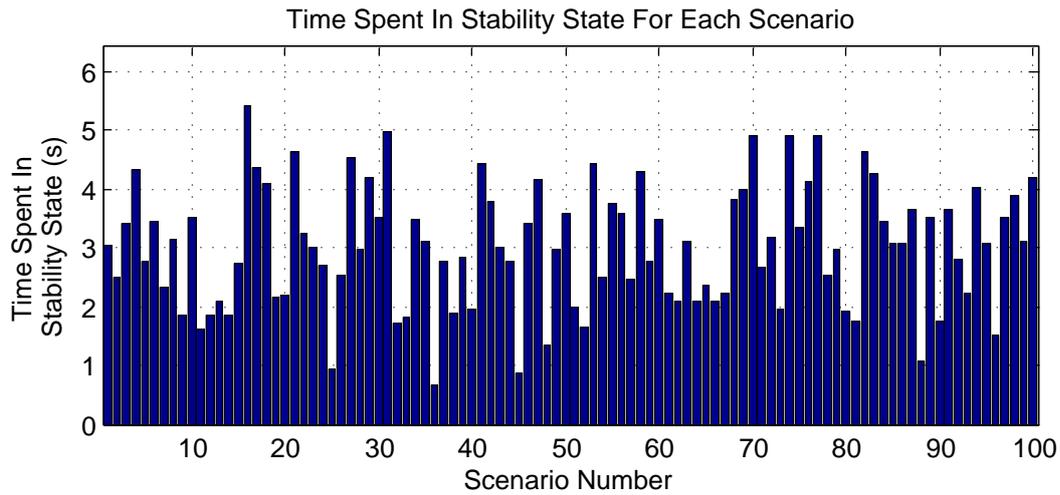
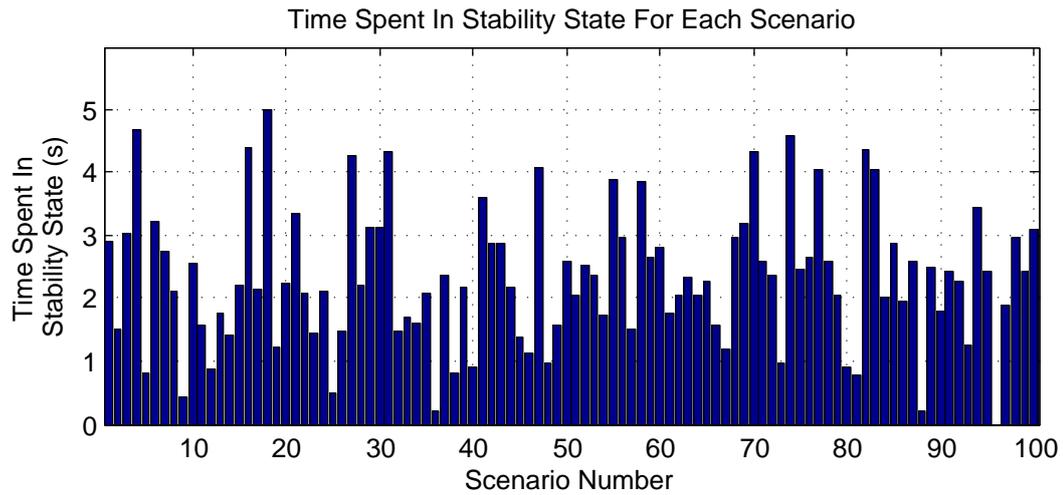
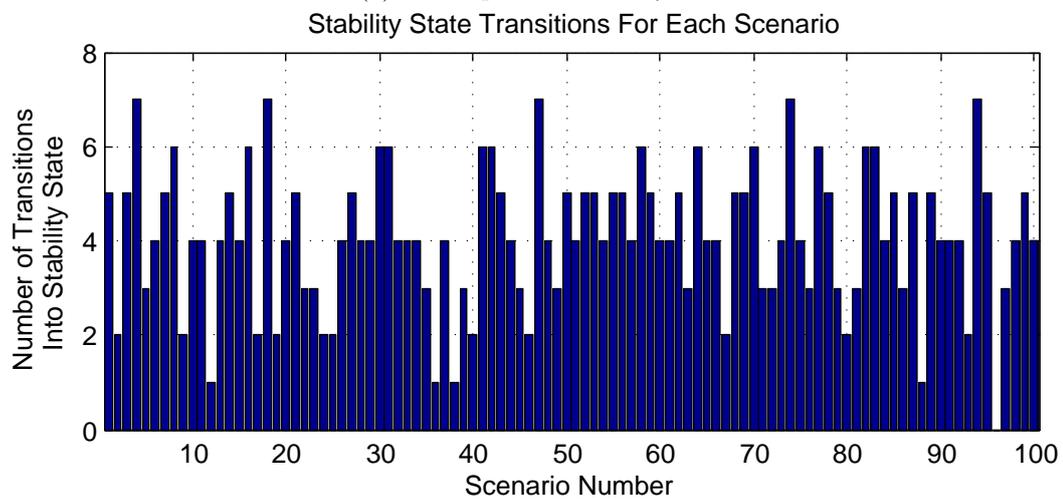


Figure 5.24: Low manipulability simulation: stability results.



(a) Time spent in stability state.



(b) Number of state transitions into stability.

Figure 5.25: High manipulability simulation: stability results.

5.4.2 No Manipulability Hysteresis

The hysteresis for the manipulability threshold is removed and its effect investigated in this section. The thresholds used for this simulation is shown in table 5.17. Results for this simulation and comparison with the baseline simulation is shown in table 5.18 and figure 5.26. As for the “no stability hysteresis” simulation, where time in stability is reduced, the time spent in the manipulability state is reduced. But unlike the “no stability hysteresis” simulation, the number of state transitions into the manipulability state does not increase, but actually decreases slightly. The same trend is observed for the standard deviation results. Mean time and state transitions into stability state also increase, following the same trend as in the “no stability hysteresis” simulation for its affect on manipulability. Standard deviation results for stability however did not follow the same trend as the mean and decreased, though only slightly. The reason the number of state transitions into manipulability does not increase is again believed to be due to the soft threshold used: once the manipulability threshold is breached it is not immediately addressed, which means it can not immediately go above the threshold and out of the manipulability state again.

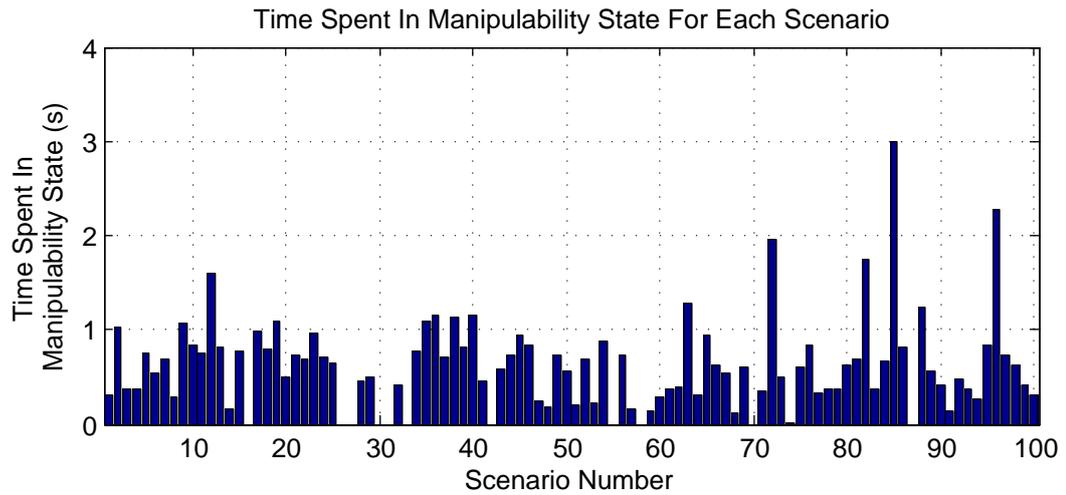
Table 5.17: No manipulability hysteresis simulation: stability and manipulability thresholds.

	Manipulability	Stability
Threshold	0.750	1.25
Hysteresis threshold	None	1.60

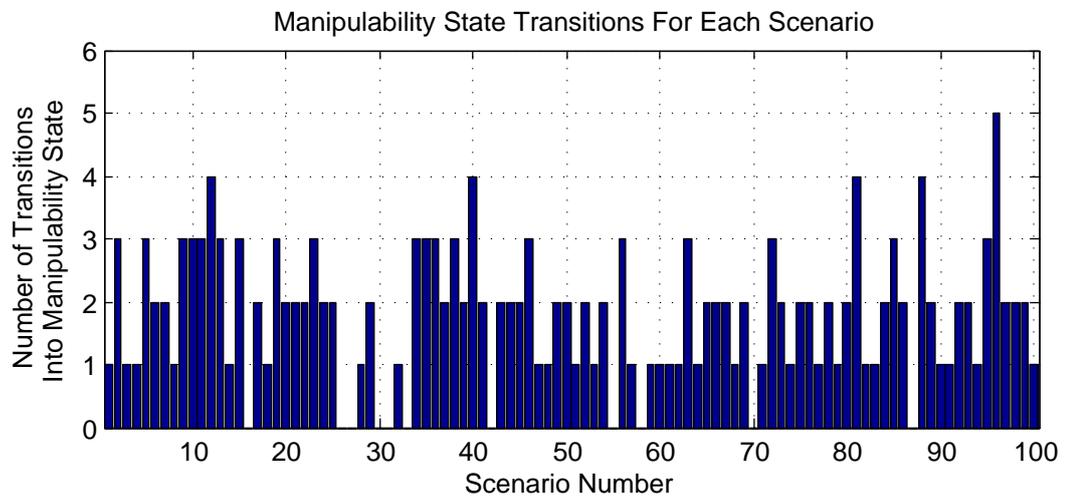
Table 5.18: No manipulability hysteresis simulation: mean and standard deviation results of all scenarios with no hysteresis for manipulability.

	Results	Comparison with baseline (mean)
No. of failed scenarios	0	0
Simulation time (s)	44.9 (4.21)	+0.0154 (+0.0343%)
Simulation speed (ms^{-1})	0.448 (0.0687)	-0.000131 (-0.0292%)
Time spent moving base (s)	25.8 (57.4%) (4.23)	-0.281 (-1.08%)
Time spent moving manipulator (s)	15.6 (34.6%) (0.870)	+0.748 (+5.05%)
Time in manipulability state (s)	0.613 (1.37%) (0.491)	-0.739 (-54.6%)
No. of manipulability state transitions	1.79 (1.07)	-0.0400 (-2.19%)
Time in stability state (s)	2.95 (6.58%) (1.07)	+0.288 (+10.8%)
No. of stability state transitions	5.21 (1.38)	+0.570 (+12.3%)

The same type of simulations was performed to see a range of manipulability threshold values with no hysteresis and their effect on the controller. The range of threshold values used is shown in table 5.19 and the results in table 5.20. Only 2 scenarios failed across the simulations, both in threshold 5 due to an infinite loop, caused by waypoints placed very close to each other and requiring a large heading change. It is not certain if



(a) Time spent in manipulability state.



(b) Number of state transitions into manipulability.

Figure 5.26: No manipulability hysteresis simulation: manipulability results.

this was due to not having any hysteresis, or just because the manipulability threshold is set too high, as the maximum manipulability metric is 1 . In either case, these results again show that manipulability has a lesser impact on controller stability compared to stability when considering extreme cases, as the number of failed scenarios is much less than the extreme stability threshold case.

Table 5.19: No manipulability hysteresis simulation: range of stability and manipulability thresholds

	Stability	Manipulability
Threshold 1	1.25	0.600
Threshold 2		0.700
Threshold 3		0.800
Threshold 4		0.900
Threshold 5		0.990
Hysteresis Threshold	1.6	None

Table 5.20: No manipulability hysteresis simulation: number of failed scenarios for each simulation

	No. of failed scenarios
Threshold 1	0
Threshold 2	0
Threshold 3	0
Threshold 4	0
Threshold 5	2

5.5 EXTENDED CONTROLLER: OBSTACLE AVOIDANCE

The operation of the proposed controller has been successfully shown in simulation and the next simulation will demonstrate the extended controller with obstacle avoidance capabilities. Four scenarios are selected, and two obstacles manually placed in the expected path of the mobile manipulator. In this section, in-depth evaluation of the performance of the obstacle avoidance controller will not be performed as the aim of adding the obstacle avoidance controller is to demonstrate the modularity and flexibility of the overall controller. This is the reason only four scenarios are selected to see if all of the sub controllers can work together.

The particular scenarios that were selected all contain paths that cross each other, with the obstacles placed in this area so it can have more impact on the scenario. The obstacle is a solid, circular column with a radius of 0.3 m and a height of 1 m. The radius is selected to create an obstacle that is of comparable size to the mobile manipulator and the height is arbitrarily selected as it does not factor in for this particular obstacle controller. Figure 5.27 shows the selected scenarios with the obstacle placements shown

as large circles. Mobile manipulator parameters used are the same ones used for the baseline simulation. The maximum time limit has been increased to two minutes to account for the extra time that will be needed to complete a scenario. The requirement for a completed scenario is also updated to include not hitting any obstacles when reaching all the waypoints.

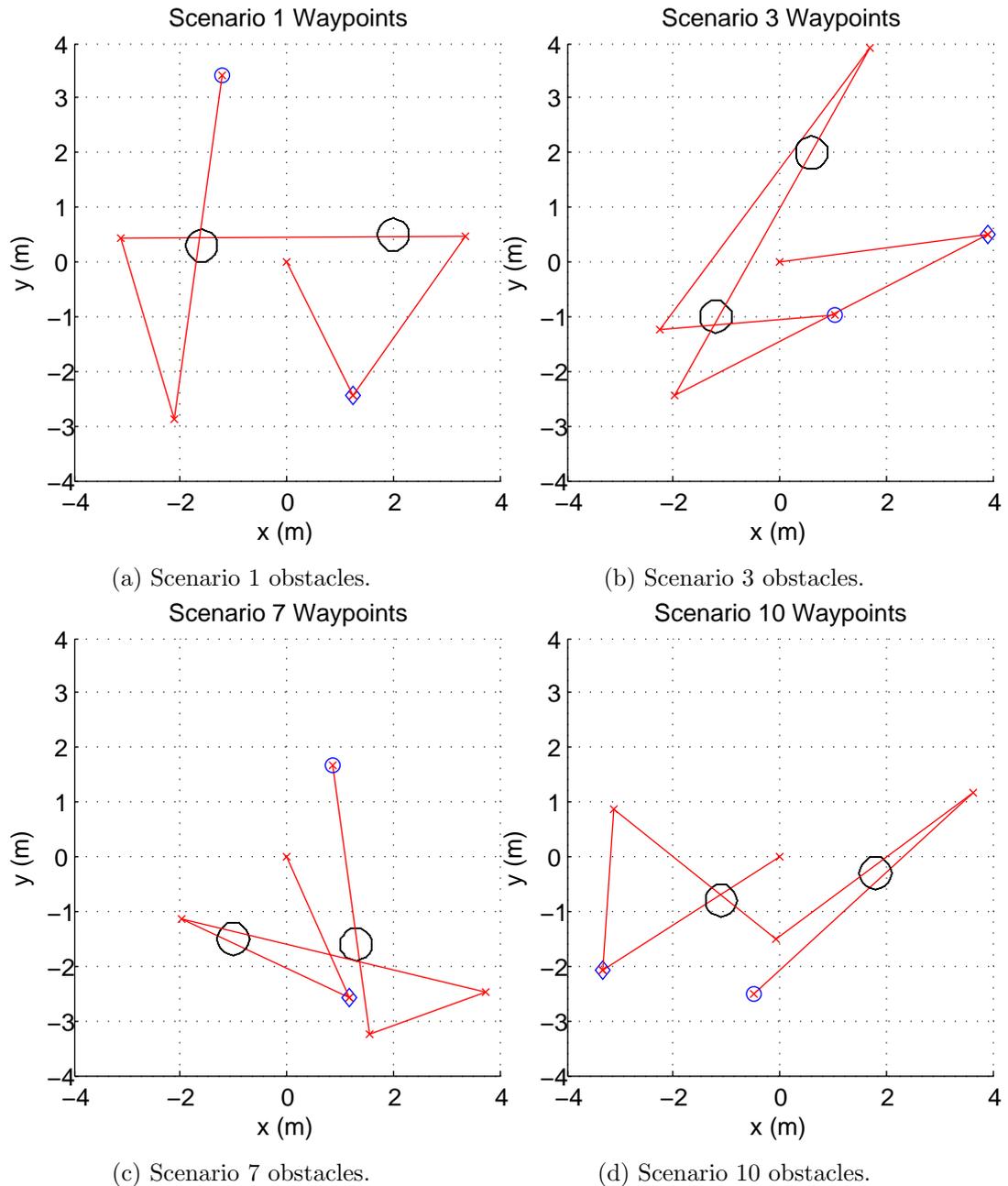


Figure 5.27: Obstacle placements in the selected scenarios.

Results of the four scenarios are presented in table 5.21 and the path taken to avoid the obstacles are shown in figures 5.28 and 5.29. The results of each scenario show that the path of the mobile base and manipulator have avoided the obstacles denoted by the

circles, and that the manipulator have reached all the waypoints denoted by the crosses. Sharp changes in the path of the manipulator indicates state changes, and can be seen occurring near the waypoints in all of the scenarios, caused by either the manipulability or stability controllers being triggered. As expected, the overall simulation time for each scenario is higher compared with the baseline due to the mobile base having to drive around obstacles. Stability and manipulability results are also slightly affected compared to the baseline, due to the obstacles changing the straight-line paths which were taken in the baseline simulation to reach each waypoint .

A high number of obstacle state transitions can be seen in scenario 7 and is due to state oscillations occurring. Figure 5.30 shows the controller states for scenario 7 which has state oscillations and scenario 10, which does not. The oscillations are caused by the way the obstacle controller defines when an obstacle is in range, and therefore, to enter into the obstacle avoidance state. Part of this definition is dependent on the current velocity of the mobile base, with the idea being a faster moving robot will require more room to manoeuvre around an obstacle. The mobile base speeds up to reach the last waypoint in scenario 7, causing the obstacle it is passing to be within the obstacle in range definition. The mobile base then slows down to begin to avoid the obstacle, causing it to be out of the obstacle in range definition. This is repeated until the mobile base is sufficiently far away from or heading in significantly different direction such that the obstacle in range detection is not triggered again. A more sophisticated obstacle avoidance controller might eliminate this problem.

Table 5.21: Performance results of each scenario with obstacle avoidance.

	Scenario			
	1	3	7	10
Scenario Complete	Yes	Yes	Yes	Yes
Simulation time (s)	52.3	62.0	55.3	64.6
Simulation speed (ms^{-1})	0.434	0.446	0.354	0.323
Time spent moving base (s)	26.5	27.3	25.3	28.6
Time spent moving manipulator (s)	14.1	15.8	14.5	16.6
Time in manipulability state (s)	0.780	0.780	1.60	1.52
No. of manipulability state transitions	1	1	2	2
Time in stability state (s)	2.98	3.08	2.78	3.34
No. of stability state transitions	5	4	5	5
Time in obstacle state (s)	7.90	15.0	11.1	14.5
No. of obstacle state transitions	5	9	26	4

The results presented here show that the extended controller with obstacle avoidance is capable of driving around simple obstacles, and still completing the task of reaching all the waypoints. The addition of the obstacle avoidance controller has not significantly affected the operation of the other controllers. Even though the obstacle controller itself is a simple implementation with room for optimisation and improve-

ment, the demonstration of the flexibility and modularity of the controller structure is achieved.

5.6 SUMMARY

A simulation setup was established to test the proposed controller in this research which consist of randomly generated scenarios containing waypoints for the mobile manipulator to reach. A set of 100 scenarios were generated and the controller tested to provide confidence that the controller is able to perform arbitrary tasks. A baseline simulation was performed using medium threshold values for the manipulability and stability metrics, with the controller successful in completing all the scenarios within the specified time limit and without breaching the stability limit. The controller was further tested by varying the threshold values of stability and manipulability, where again, the controller was able to complete all the scenarios. Failed scenarios occurred in simulations involving no threshold hysteresis and high values of stability or manipulability threshold values. The main cause of the failed scenarios was not completing the scenarios within the specified time limit. Stability was consistently shown to have a larger impact on controller performance compared to manipulability across many different simulations, which provides a starting point for any controller optimisations. The modularity of the controller was demonstrated by showing obstacle avoidance results, where obstacles were placed in four scenarios and required the mobile manipulator to move around them to reach the waypoints. The controller was able to reach all the waypoints in the scenarios and avoid all obstacles while still maintaining the performance metrics.

The thesis is concluded in Chapter 6 which provides a summary of the work that has been done in the design and implementation of the proposed controller.

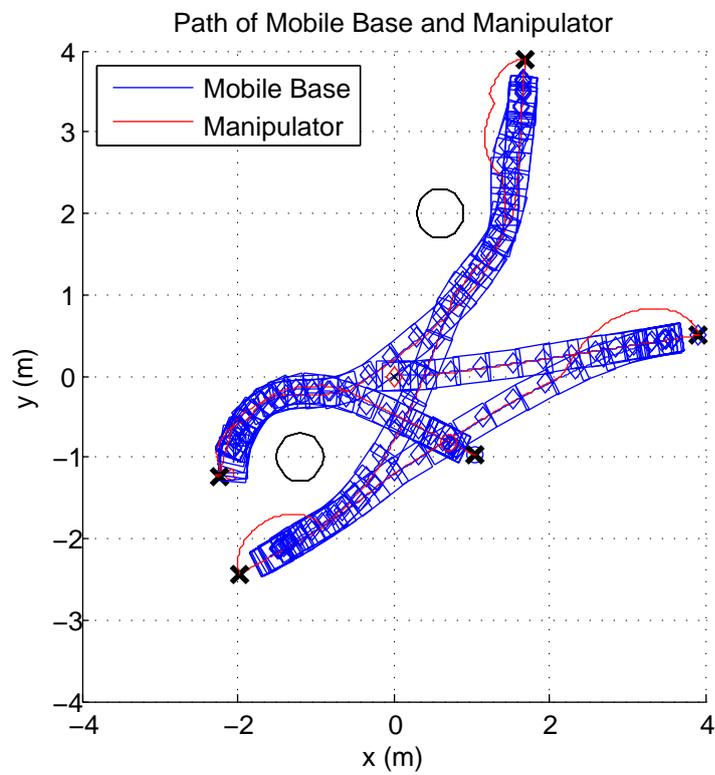
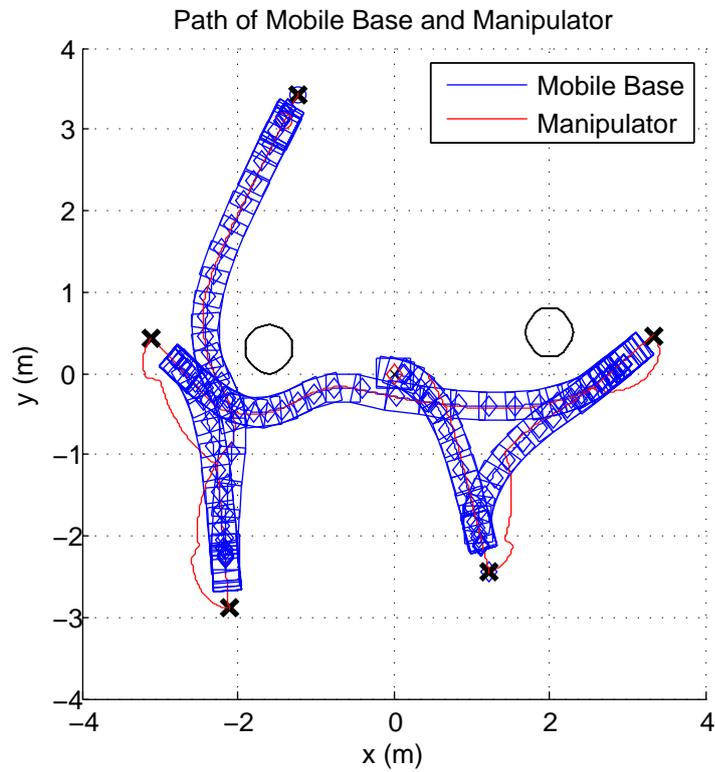
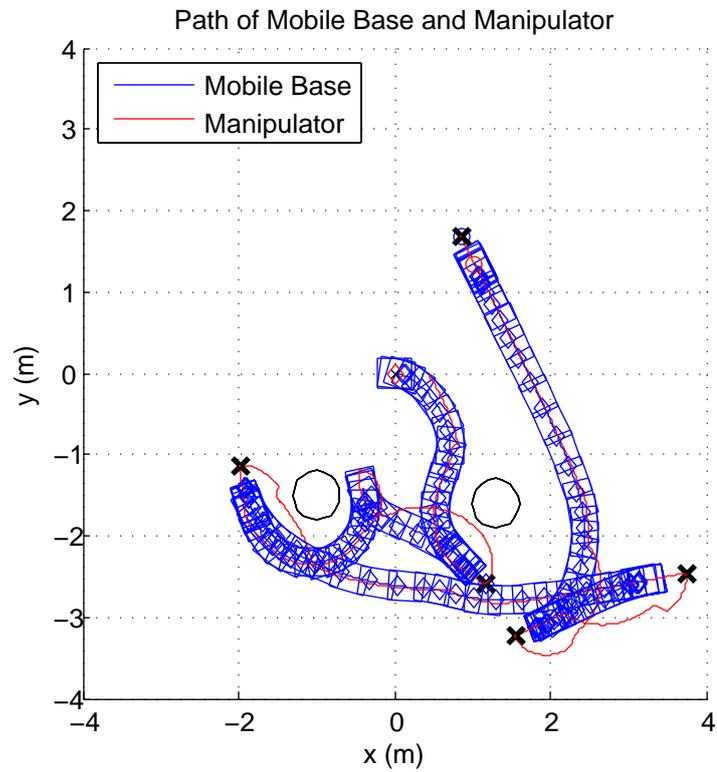
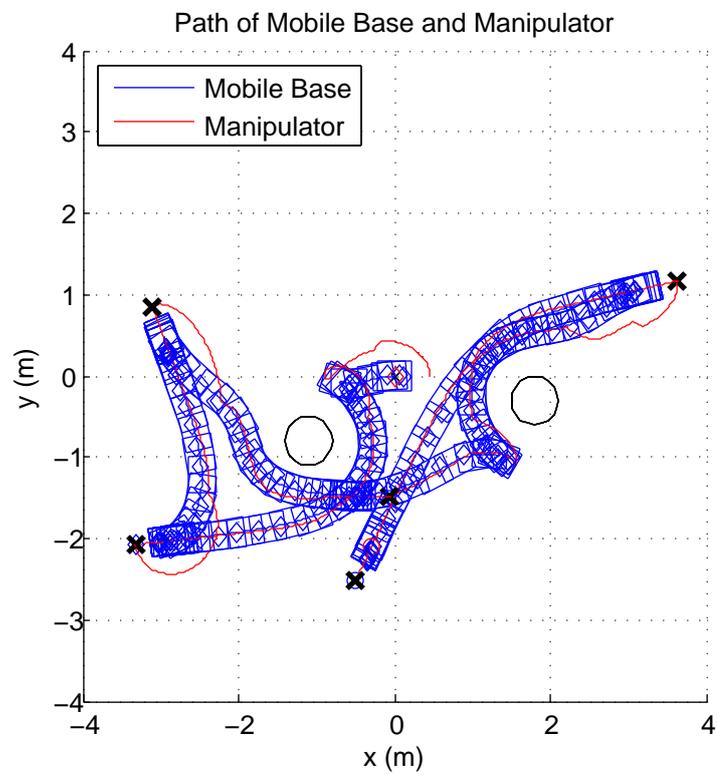


Figure 5.28: Path taken by mobile base and manipulator for scenarios 1 and 3.

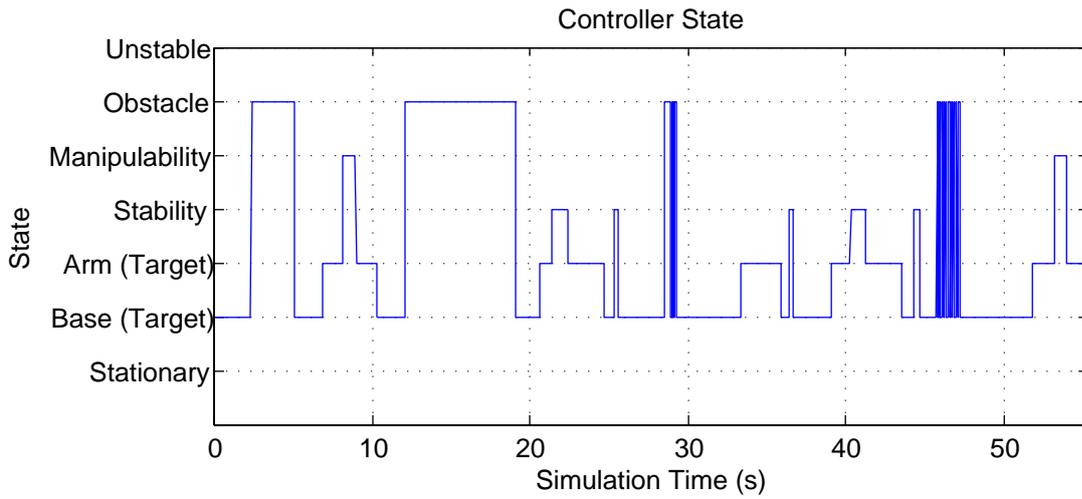


(a) Scenario 7.

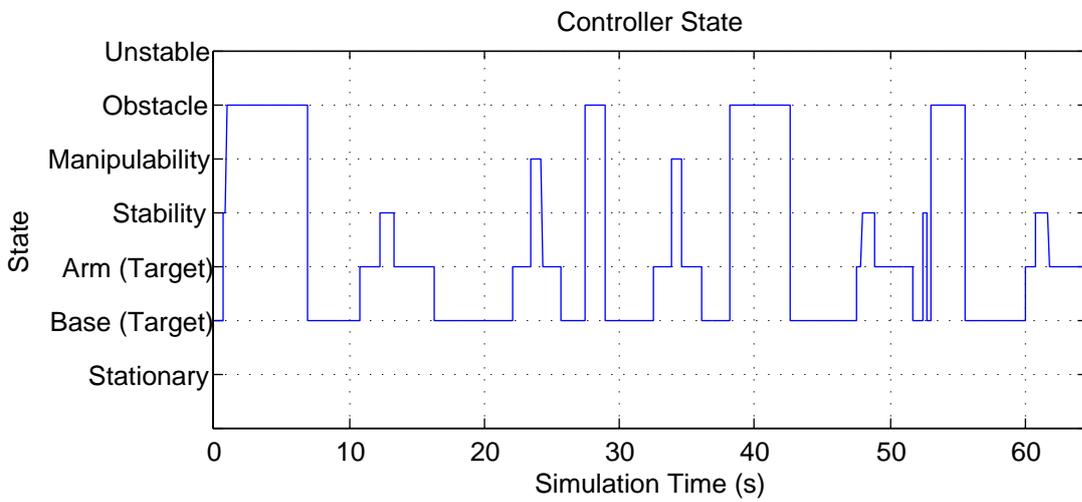


(b) Scenario 10.

Figure 5.29: Path taken by mobile base and manipulator for scenarios 7 and 10.



(a) Scenario 7.



(b) Scenario 10.

Figure 5.30: Controller state of the mobile manipulator during scenario 7 and 10.

Chapter 6

CONCLUSION

This final chapter concludes this research and discusses the results and limitations of the proposed controller. A summary of the work that has been performed is first presented, as well as the main outcomes of the thesis. The strengths and limitations of the proposed controller are then discussed and future work opportunities that can improve or take advantage of the controller are outlined.

6.1 THESIS SUMMARY

In this thesis, a modular, behaviour-based, hierarchical controller was proposed for the control system of a mobile manipulator. The aim of this approach was to create a controller that was flexible, easy to understand and implement. This is in contrast to monolithic approaches that have been used in previous research. The application of behaviour-based robotics, a simple concept, to the complex system of a mobile manipulator is demonstrated to prove that one does not need a complex controller to control a mobile manipulator system.

Behaviour-based controllers are inherently modular, as the individual behaviours form a sensorimotor pair where a simple, predefined action is performed for a certain input. Behaviour-based controllers are also purely reactive, which eliminates the need for planning and the complexities associated with performing planning operations. A hierarchical controller structure was used to organise and coordinate behaviours to perform more complex tasks.

The task for the controller to perform consists of reaching a series of waypoints with the end effector. Two performance metrics, manipulability and tip-over stability were used to evaluate the performance of the controller. Three behaviour-based controllers were developed to perform this task: objective controller, manipulability controller and stability controller. The objective controller performs the actual task of reaching the waypoints, while the manipulability and stability controller maintain their respective performance metric above a certain threshold value during the task execution process. A fourth, obstacle avoidance controller was added to the system

to operate in conjunction with the existing controllers to demonstrate the modularity and ease of expansion of the proposed controller. Finally, a state machine was used to coordinate the controllers and determine which should be executed when conditions satisfy the execution criteria for multiple controllers.

A mobile manipulator model and simulation environment was developed to test the controller using randomly generated scenarios containing randomly generated way-points, for a range of performance metric threshold values. The controller was capable of completing the majority of the scenarios in a timely manner while maintaining manipulability and stability above their respective threshold values. Failed scenarios occurred only when very high stability and manipulability threshold values were used without hysteresis. The failing condition for those scenarios were all time constraint breaches, and not manipulability or stability threshold breaches. Stability was found to have a larger impact on controller performance than manipulability.

The controller proposed in this thesis is specifically targeted at performing the key tasks of a mobile manipulation system, that of maintaining manipulability and stability while still executing the required task. This research has shown that a hierarchical, behaviour-based approach not only works, but has many advantages compared to traditional robot controller structures. This research provides a controller framework as well as controllers targeting the key tasks in mobile manipulation that future research can use and expand upon.

6.2 LIMITATIONS AND FUTURE WORK

Limitations of the proposed controller are that it has only been simulated in a static environment without additional external disturbances. The controller also uses exact positioning information with respect to the mobile base and manipulator. In real world scenarios, there inevitably will be external disturbances such as uneven work surfaces and position errors due to sensor noise. This is not to say the controller will not work when external disturbances or sensor noise is added, but further experiments can be done to determine their effects on controller performance.

The current controller has only been tested in a simulation environment. An avenue for future research is the implementation of the controller on an actual mobile manipulator system. Implementing the controller on a real system also investigates the limitations described earlier. Not only can testing on a real system further validate the controller, any performance discrepancies between simulation and actual tests can provide further insights for the mobile manipulator system model that was developed, or how some controllers behave in a slightly different way, which can be used to further improve the simulation environment and controller.

Another prospect for future work is expanding or altering the capabilities of the controller by taking advantage of the modular controller design. This can be either

adding new behaviours to the controller or changing the execution and coordination of behaviours. Examples of new behaviours are a pick-and-place controller or an end effector trajectory tracking controller. The PD movement controller or state machine coordinator can also be changed to alter the performance characteristic; for example, using an adaptive PD controller to tune the movement controllers gains to better account for unknown or inaccurate system or external dynamics.

REFERENCES

- ADASCALITEI, F. AND DOROFTEI, I. (2011), ‘Practical applications for mobile robots based on mecanum wheels - a systematic survey’, In *International Conference On Innovations, Recent Trends And Challenges In Mechatronics, Mechanical Engineering And New High-Tech Products Development*.
- ARKIN, R.C. (1998), *Behavior-Based Robotics*, MIT Press, Cambridge, MA.
- ARKIN, R. AND MACKENZIE, D. (1994), ‘Planning to behave: A hybrid deliberative/reactive robot control architecture for mobile manipulation’.
- BØGH, S., HVILSHØJ, M., KRISTIANSEN, M. AND MADSEN, O. (2012), ‘Identifying and evaluating suitable tasks for autonomous industrial mobile manipulators (aimm)’, *The International Journal of Advanced Manufacturing Technology*, Vol. 61, pp. 713–726.
- BROOKS, R. (1986), ‘A robust layered control system for a mobile robot’, *Robotics and Automation, IEEE Journal of*, Vol. 2, No. 1, mar, pp. 14 – 23.
- CARRIKER, W., KHOSLA, P. AND KROGH, B. (1989), ‘An approach for coordinating mobility and manipulation’, In *Systems Engineering, 1989., IEEE International Conference on*, pp. 59 –63.
- CARRIKER, W., KHOSLA, P. AND KROGH, B. (1991), ‘Path planning for mobile manipulators for multiple task execution’, *Robotics and Automation, IEEE Transactions on*, Vol. 7, No. 3, jun, pp. 403 –408.
- CHEN, Y., LIU, L., ZHANG, M. AND RONG, H. (2006), ‘Study on coordinated control and hardware system of a mobile manipulator’, In *Proceedings of the 6th World Congress on Intelligent Control and Automation*,.
- CHRISTENSEN, H.I. (2009), ‘A roadmap for us robotics - from internet to robotics’, Report. Workshop on Emerging Technologies and Trends.
- CORKE, P. (2008), ‘Robotics toolbox for matlab’, Website. http://www.petercorke.com/Robotics_Toolbox.html.

- CORKE, P. AND ARMSTRONG-HÉLOUVRY, B. (1995), ‘A meta-study of PUMA 560 dynamics: A critical appraisal of literature data’, *Robotica*, Vol. 13, No. 3.
- CRESPI, V., GALSTYAN, A. AND LERMAN, K. (2005), ‘Comparative analysis of top-down and bottomup methodologies for multiagent system design’, In *Forth International Join Conference On Autonomous Agents And Multiagent Systems*.
- DASSAULT SYSTÉMES (2012), ‘3DS’, Website. <http://www.3ds.com>.
- DENAVID, J. AND HARTENBERG, R.S. (1955), ‘A kinematic notation for lower-pair mechanisms based on matrices’, *Trans ASME J. Appl. Mech*, Vol. 23, pp. 215–221.
- DENSO (2009), ‘Packaging automation trends: using small assembly robots in upstream packaging processes’. <http://www.processonline.com.au/articles/36410-Packaging-automation-trends-using-small-assembly-robots-in-upstream-packaging-processes>.
- DUBOWSKY, S. AND VANCE, E. (1989), ‘Planning mobile manipulator motions considering vehicle dynamic stability constraints’, In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, may, pp. 1271–1276 vol.3.
- ELGAWI, O. (2009), ‘Architecture of behavior-based and robotics self-optimizing memory controller’, In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, may, pp. 3712–3717.
- ELLEKILDE, L.P. AND JORGENSEN, J.A. (2010), ‘Robwork: A flexible toolbox for robotics research and education’, *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, Vol. , No. , june, pp. 1–7.
- ERBATUR, K., OKAZAKI, A., OBIYA, K., TAKAHASHI, T. AND KAWAMURA, A. (2002), ‘A study on the zero moment point measurement for biped walking robots’, In *Advanced Motion Control, 2002. 7th International Workshop on*, pp. 431–436.
- FIRBY, R. (1987), ‘An investigation into reactive planning in complex domains’, In *Proceedings of The Sixth National Conference On Artificial Intelligence*.
- FURUNO, S., YAMAMOTO, M. AND MOHRI, A. (2003), ‘Trajectory planning of mobile manipulator with stability considerations’, In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, sept., pp. 3403–3408 vol.3.
- GAT, E. (1998), ‘On three-layer architectures’, In *Artificial Intelligence and Mobile Robots*, MIT Press.

- GONG, K. AND MCINNES, A. (2011), ‘A hierarchical control scheme for coordinated motion of mobile manipulators’, In *Automation, Robotics and Applications (ICARA), 2011 5th International Conference on*, pp. 115–120.
- GONG, K. AND MCINNES, A. (2013), *Recent Advances in Robotics and Automation*, Vol. 480 of Studies in Computational Intelligence, Springer-Verlag, Chap. A Modular Hierarchical Control Scheme for Mobile Manipulation.
- HAMNER, B., KOTERBA, S., SHI, J., SIMMONS, R. AND SINGH, S. (2010), ‘An autonomous mobile manipulator for assembly tasks’, *Autonomous Robots*, Vol. 28, No. 1.
- HUANG, Q., SUGANO, S. AND TANIE, K. (1998), ‘Motion planning for a mobile manipulator considering stability and task constraints’, In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, may, pp. 2192–2198 vol.3.
- HUANG, Q., TANIE, K. AND SUGANO, S. (2000), ‘Coordinated motion planning for a mobile manipulator considering stability and manipulation’, *The International Journal of Robotics Research*.
- HVILSHØJ, M., BØGH, S., MADSEN, O. AND KRISTIANSEN, M. (2009), ‘The mobile robot little helper: Concepts, ideas and working principles’, In *Emerging Technologies Factory Automation, 2009. ETFA 2009. IEEE Conference on*, sept., pp. 1–4.
- IAGNEMMA, K., RZEPNIEWSKIA, A., S. DUBOWSKYA, P.P., HUNTSBERGERB, T. AND SCHENKER, P. (2000), ‘Mobile robot kinematic reconfigurability for rough-terrain’.
- INTERNATIONAL FEDERATION OF ROBOTICS (2012), ‘History of industrial robots’, Brochure. http://www.ifr.org/uploads/media/History_of_Industrial_Robots_online_brochure_by_IFR_2012.pdf.
- IROBOT (2012), ‘Roomba’, Website. <http://www.irobot.com/>.
- JITENDRA, J. AND RAOL, R. (2012), *Mobile Intelligent Autonomous Systems*, CRC Press.
- KATZ, D., HORRELL, E., YANG, O., BURNS, B., BUCKLEY, T., GRISHKAN, A., ZHYLKOVSYY, V., BROCK, O. AND LEARNED-MILLER, E. (2006), ‘The umass mobile manipulator uman: An experimental platform for autonomous mobile manipulation’, In *In Workshop on Manipulation in Human Environments at Robotics: Science and Systems*.

- KHATIB, O. (1985), ‘Real-time obstacle avoidance for manipulators and mobile robots’, In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, mar, pp. 500 – 505.
- KOLMANOVSKY, I. AND MCCLAMROCH, N. (1995), ‘Developments in nonholonomic control problems’, *Control Systems, IEEE*, Vol. 15, No. 6, dec, pp. 20 –36.
- KOTHA, S. (1995), ‘Mass customization: Implementing the emerging paradigm for competitive advantage’, *Strategic Management Journal*, Vol. 16, No. S1, pp. 21–42.
- KOZLOWSKI, K., DUTKIEWICZ, P. AND WRBLEWSKI, W. (2003), ‘Modeling and control of robots’.
- LEWIS, F.L., DAWSON, D.M. AND ABDALLAH, C.T. (2004), *Robot Manipulator Control Theory and Practice*, Marcel Dekker.
- LIN, S. (2001), *Robust and Intelligent Control of Mobile Manipulators*, PhD thesis, University of Toronto.
- MACKENZIE, D. AND ARKIN, R. (1996), ‘Behavior-based mobile manipulation for drum sampling’, In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, apr, pp. 2389 –2395 vol.3.
- MOBILEROBOTS, A. (2012), ‘Pioneer P3-DX’, Website. <http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx>.
- NAGATANI, K., HIRAYAMA, T., GOFUKU, A. AND TANAKA, Y. (2002), ‘Motion planning for mobile manipulator with keeping manipulability’, In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, pp. 1663 – 1668 vol.2.
- NASA (2010), ‘Mars rover’, Website. <http://marsrover.nasa.gov>.
- OGREN, P., EGERSTEDT, N. AND HU, X. (2000), ‘Reactive mobile manipulation using dynamic trajectory tracking’, In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, pp. 3473 –3478 vol.4.
- PAPADOPOULOS, E. AND REY, D. (1996), ‘A new measure of tipover stability margin for mobile manipulators’, In *Proceedings of the IEEE International Conference On Robotics and Automation*.
- RAY, D., MANDAL, A., MAJUMDER, S. AND MUKHOPADHYAY, S. (2011), ‘Human-like gradual multi-agent q-learning using the concept of behavior-based robotics for autonomous exploration’, In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, dec., pp. 2725 –2732.

- ROBOCUP 2009 (2009), ‘Robocup2009’, Website. <http://www.robocup2009.org/>.
- ROSHEIM, M. (2006), *Leonardos Lost Robots*, Springer.
- SHAMAH, B. (1999), *Experimental Comparison of Skid Steering Vs. Explicit Steering for a Wheeled Mobile Robot*, Master’s thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, March.
- SINGH, H. AND SUKAVANAM, N. (2011), ‘Neural network based adaptive compensator for motion/force control of constrained mobile manipulators with uncertainties’, In *Hybrid Intelligent Systems (HIS), 2011 11th International Conference on*, pp. 253–258.
- SPONG, M.W. AND FUJITA, M. (2011), ‘Control of robotics’, Report. The Impact of Control Technology, T. Samad and A.M. Annaswamy (eds.), 2011. Available at www.ieeecss.org.
- TAKANISHI, A., OK LIM, H., TSUDA, M. AND KATO, I. (1990), ‘Realization of dynamic biped walking stabilized by trunk motion on a sagittally uneven surface’, In *Intelligent Robots and Systems ’90. ’Towards a New Frontier of Applications’, Proceedings. IROS ’90. IEEE International Workshop on*, jul, pp. 323–330 vol.1.
- TELEROB (2010), ‘Telerob’, Website. <http://www.telerob.de/teodor>.
- WASIK, Z. (2004), *A Behavior-Based Control System for Mobile Manipulation*, PhD thesis, Orebro University.
- YAMAMOTO, Y. AND YUN, X. (1994), ‘Modeling and compensation of the dynamic interaction of a mobile manipulator’, In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, may, pp. 2187–2192 vol.3.
- YAMAMOTO, Y. AND YUN, X. (1995), ‘Coordinating locomotion and manipulation of a mobile manipulator’, *IEEE Transactions on Automatic Control*, Vol. 39, No. 6, June.
- YOSHIKAWA, T. (1990), *Foundations of Robotics: Analysis and Control*, MIT Press.