The Pennsylvania State University
The Graduate School

A COUPLED MOBILE BASE MANIPULATION SYSTEM FOR ROBOTIC REFUELING

A Thesis in
Mechanical Engineering
by
Kathryn Briggs

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

August 2014

The thesis of Kathryn Briggs was reviewed and approved* by the following:

Sean Brennan
Associate Professor of Mechanical Engineering
Thesis Advisor

Karl Reichard
Research Associate and Assistant Professor of Acoustics, Applied Research Laboratory
Thesis Co-Advisor

H. J. Sommer III
Professor of Mechanical Engineering
Reader

Karen A. Thole
Professor of Mechanical Engineering
Department Head of Mechanical and Nuclear Engineering

*Signatures are on file in the Graduate School.

Abstract

Technological advancements in computer vision, path planning, and spatial awareness have enabled the development of robotic manipulation systems to perform elementary dexterous tasks. A specific application considered here consists of having a robotic manipulator refuel a vehicle gas tank. Currently there are no standard mobile robotic refueling systems. Moreover, although increasingly more common, ground robots have not been used specifically for refueling tasks. This thesis examines the ability of a three-degree-of-freedom robotic manipulator coupled with a mobile base to perform a vehicle refueling task. A model of a three-degree-of-freedom robotic manipulator was fully implemented with a GAZEBO simulation package to test the refueling process in a virtual world. This allowed virtual testing of multiple vehicles and fuel tank configurations. Tests were then conducted using a real three-degree-of-freedom robotic manipulator, a RE2 Automatic Arm, fully integrated with ROS (Robot Operating System). For angled fuel tank inlets, an additional degree of freedom was required, where a wrist joint was needed to properly align the nozzle along the central axis of the angled fuel tank inlet tube. The advantage of using a mobile base as an additional degree of freedom during refueling is assessed for a sample refueling trajectory. Specific geometries of the robotic manipulator in relation to the mobile base were incorporated into an analysis of the optimal base location for each trajectory point along the desired refueling path. Through the investigation of mobile base and manipulator movement, specific constraints are quantified for the prototype design of a wheeled mobile base. The analysis results also provide insight into general design principles for robotic refueling systems.

Table of Contents

List of Figures

List of Tables

Acknowledgments

This thesis is based upon work supported by the Army RDECOM-TARDEC project under Contract Number N00024- l 2-D-6404, Delivery Order Number 0111. The content of the information does not necessarily reflect the position or policy of the Government, and no official endorsement should be inferred.

I would like to express my gratitude to my thesis advisor Dr. Sean Brennan for continuous guidance throughout my undergraduate and graduate degrees. Your encouragement has allowed me to grow as an engineer, and pursue my passion in robotics. I would also like to thank my thesis co-advisor Dr. Karl Reichard for your support throughout my education at Penn State. Your insight has been invaluable. I would like to thank Dr. Sommer for all the knowledge you passed on in your classes, and for your valuable comments and suggestions with my thesis. In addition, I would also like to thank Jesse Pentzer, for your selfless assistance and advice throughout the entirety of my graduate degree.

Finally I would like to thank my friends and family for the words of encouragement and love that have given me strength throughout my education.

Chapter 1

Introduction

1.1 Introduction

Recent technological advancements in computer vision, path planning, and spatial awareness has allowed for robotic manipulation systems to perform various dexterous tasks. With these advancements in technology, there has been interest in using robotic manipulation systems to complete elementary tasks specific for military base operation. One such task that is considered in this thesis is robotic vehicle refueling. For this task, a robotic manipulator is assessed to semi-autonomously insert a nozzle into a fuel tank inlet to refuel a vehicle. Vehicle refueling is a daily necessity, and can become quite difficult in remote areas, areas with rough terrain, or harsh environments where human exposure must be limited. Refueling difficulties grow in military applications in high-risk environments where the opportunity to stop and refuel is minimal or dangerous. Furthermore, fully autonomous convoy operations will require similarly autonomous refueling and support functions independent of human activity. Currently there are no standard mobile robotic refueling systems. Moreover, although increasing in availability, ground robots have not been used specifically for refueling tasks. This thesis seeks to explore the use of ground vehicles with robotic manipulators for vehicle refueling. There are several high-level tasks required to refuel a vehicle or piece of stationary equipment. These tasks are:

1. Navigate nearby, and identify the fuel inlet on a vehicle or piece of equipment

2. Attach electrical grounding system if necessary
3. Position the robotic refueling vehicle relative to the equipment (or conversely, position the equipment relative to the refueling robotic vehicle)
4. Remove fuel inlet cover or cap
5. Move fuel nozzle from stowed position to fuel inlet
6. Insert nozzle in fuel inlet
7. Dispense fuel
8. Remove nozzle from fuel inlet
9. Return nozzle to stowed position
10. Replace fuel inlet cover or cap
11. Remove grounding system

Execution of each of the high-level tasks may require a combination of sensing, control and actuation. We assume hereafter that the refueling system consists of a manipulator arm with several degrees of freedom (revolute joints) and a wheeled mobile base or some other means of positioning the manipulator arm relative to the fuel inlet. For this thesis, we are operating under the assumption that the robotic manipulator has identified the vehicle fuel inlet, and navigated to the fuel tank so that the fuel inlet is inside the manipulator's workspace.

The positioning task requires that a wheeled mobile base is best positioned to allow the robotic manipulator to align the fuel nozzle with the axis of the filler tube, and insert the end of the nozzle into the filler tube. In addition to allowing proper nozzle alignment, the manipulator must also avoid collisions with any overhanging or otherwise interfering structural elements on or around the vehicle/equipment. A comparison of several military vehicles in this thesis shows that the approach to the fuel filler tube is unrestricted in some cases and restricted in others. For example, on some tactical wheeled vehicles, such as the military equivalent of long-haul tractors (M915/M916), there may be a step or part of the cab above the filler tube, thereby restricting direct access from above. In the case of the Stryker, the

material surrounding the rear hatch protrudes on one side of the filler tube. In the case of the tactical quiet generator, the filler tube is located in a rectangular-shaped recess.

While standardization of the filler tube size and length would simplify the design of the robotic refueling system, and may be feasible, it does not seem reasonable that each vehicle would be modified to provide unrestricted access to the filler tube from all approach angles; hence path planning and base location are critical sub functions of the nozzle-positioning task. Standardization of the filler tube angle is irrelevant since the entry angle for placement of the nozzle also depends on the relative pitch of the vehicle and refueling station. The relationship between the trajectory path and the arm and base location is the primary focus within this thesis, applying this insight into planning and coordinating movement for a refueling task. Results from this study will also provide design specifications for building a prototype mobile base for a refueling task.

The removal and return of the fuel inlet cap may be the most difficult manipulation tasks, and may be the requirements that drive the design and cost of the system. In most military systems, the fuel inlet is covered by a threaded cap that must be screwed off then on to the matching threaded end of the filler tube. As anyone who has attempted to place a nut on a threaded bolt, attach a hose to a faucet, or perform any number of similar tasks knows, this seemingly simple task requires fairly precise alignment of the threaded cap (nut, hose end, etc.) and actuator (the hand and arm) with the axis of the other threaded part to successfully tighten the joint without cross-threading the two pieces and causing them to bind. The difficulty of this task is demonstrated in the recent DARPA Robotics Challenge in which one of the tasks required the robots to attempt to attach a fire hose to a service Y as shown in Figure 1.1.

Because the kinematics of this task are fundamentally different from those required to position and insert the fuel filler nozzle in the inlet tube, successfully executing the complete refueling process may require two separate manipulator arms: one for filler cap removal/replacement and another for the actual nozzle insertion and removal. An alternative solution is to use a single arm with interchangeable end effectors capable of accomplishing these different tasks. Because the removal of the fuel cap is fundamentally challenging for a robotic manipulator

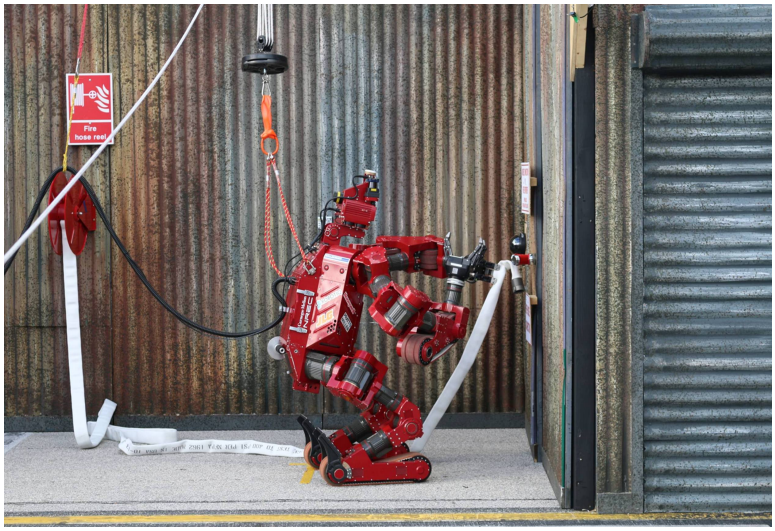

Figure 1.1. Fire hose task from DARPA Robotics Challenge. Photo credit Dr. Karl Reichard.

to perform a refueling task, further study could provide insight on fuel tank cap modifications that could greatly facilitate robotic vehicle refueling. For the purpose of this study, all simulated refueling trajectories assume the fuel cap has been removed prior to refueling.

Actual dispensing of the fuel can be very simple if it is assumed that the fuel dispensing system includes internal valves to control the flow of fuel. This would eliminate the need to design a robot capable of grasping and squeezing a handle to control the flow of the fuel, as done in most manual refueling stations. This approach is recommended and provides additional levels of safety and the possibility of designing both hardware and software interlocks to avoid accidental fuel dispensing. The removal of the fuel nozzle is also very simple. If the nozzle has been successfully inserted into the tank, the manipulator will simply follow its insertion trajectory in reverse to remove the nozzle from the fuel tank.

For this thesis, a primary emphasis is placed on the third task, of properly positioning the wheeled mobile base and robotic manipulator to successfully carry out a refueling task. By testing a variety of mobile base positions and manipulator configurations relative to the trajectory of the end effector, insight can be drawn on how to best design a mobile refueling system. The mobile manipulator performed several refueling trajectories that simulated nozzle insertion along fuel tank inlet

angles of 90 and 45 degrees relative to the horizontal. By varying the base location of the manipulator for these tasks, optimal base positions could be chosen that minimized various metrics such as joint movement, overall joint velocity, or power consumption of the manipulator during the refueling task.

Chapter 2

Related Work

Due to the complex nature of a refueling task, it is likely that a mobile platform equipped with a robotic manipulator is a necessity to carry out the refueling process. In using a mobile manipulation system, sophistication in coordinating both the arm and base movement is required. The first step of a mobile manipulation task requires planning of both base and arm trajectories. Movement of the base and robotic arm is task specific, requiring a calculated analysis of the surrounding area. One approach taken by Stulp et al. was to quantify a probability distribution of the surrounding space of the mobile robot [4]. An action-related concept of the space was used for least commitment planning. Based upon several metrics, the probability distribution gave a best guess for where the robot could park itself to perform a task [5], [4], [6]. A look at the manipulability and optimal dexterity of a robotic arm was considered by Zacharias et al., creating a capability map for the arm based upon its base position [7]. The planning phase also takes into consideration the types of objects to be grasped. Gaschler looks specifically at the volumes of the object to be grasped to provide foresight on the method of grasping an object, based upon the robots understanding of the objects volume [8]. These methods [5],[4],[7],[6],[8] consider the mobile base and robotic manipulator as separate entities, performing a park and grasp approach. This concept is taken a step further in this study, to consider the base location as an adjustable degree of freedom while navigating along a trajectory, and thereby optimizing the base location relative to individual trajectory points.

In refueling robots, there may be a need to move the base while simultaneously

moving the arm. Base and arm motion was considered as a single coordinated activity by Berenson, taking the grasping motion into account when planning the base location and trajectory path [9]. However, the initial and final goal points were only considered when analyzing base location. Path planning was used to connect the initial and final goals. The states between the initial and final goals were not taken into account for the base analysis. A visual servoing control was employed for the grasping task by Wang et al. to maintain the goal object within the field of view of the camera using a learning algorithm [9]. A similar method to the visual servoing approach was used by Bley et al. where, instead of relying on accurate 3D images, he formulated a general object description used for planning grasping tasks [10]. These methods [9], [10] strictly rely on camera feedback for placement of the mobile base rather than an analysis the manipulator geometries with respect to the surrounding space. Camera feedback is very helpful for real-time corrections of small errors in the end effector location as it performs a task. Camera feedback however is not useful for predicting and planning where the robot will navigate. A combination of both camera feedback and an internal algorithmic assessment of manipulator geometries will optimize the robots performance during a manipulation task.

Path planning has been a source of much work in recent years, from a potential field approach [11], [12] to a set-oriented optimal path planning in [13], where a desired goal configuration is achieved through a polynomial root-finder solving for the most optimal solution. For the refueling application, the path is considered pre-determined for each vehicle, eliminating the need to find an optimal path to approach the refueling inlet.

In addition to a detailed planning phase, mobile robots must have robust navigation and control techniques to guide the robot to a desired location. The stabilization of mobile manipulators was addressed by Colbaugh [14]. With an uncertain dynamic model of a mobile manipulator, a homogeneous stabilizer based upon the kinematic system with adaptive methods could successfully control the wheeled mobile robot with a Puma-like arm [14]. A study on a coordinated control approach was also conducted by Seraji where the mobility of the base and arm was considered with equal weighting [15]. While these control schemes establish a framework for a mobile manipulation system, they do not take into account the

surrounding environment as factors within their control architecture.

Experimental metrics have also been considered when quantifying a mobile system's level of manipulability for tracked mobile robots. For example, in [16], a series of tasks were considered in measuring the robot's dexterity and mobility. This article complements the refueling study with its look at hybrid mobile robot (HMR) design based upon the hybridization of the mobile platform and manipulator arm re-designed as a single entity. This approach was purely experimental, with a control stick navigating the system by an outside operator, lacking closed loop control and semi-autonomy with the mobile robot. The kinematics of a mobile manipulator was explored by Gardener et al. when selecting the location where a robot will sit on a platform to provide highway maintenance while maintaining a reasonable base velocity [17].

This thesis explores a concept to prototype and design a mobile robot specific to a refueling task. When approaching the design of the robotic refueling system, there are several engineering and economic tradeoffs that can be considered. Minor modifications to the vehicle tank may be cost effective, allowing for less sophistication in the manipulator. More significant modifications to the vehicles, to provide uniformity across all targeted assets, may however, be too costly. However a highly sophisticated manipulation system may be desired for reliable performance and the ability to adapt to any vehicle. While performing manipulation tasks with the robotic arm, a look at the advantages and disadvantages of these tank modifications will be considered to facilitate performance.

Chapter 3

Objectives

3.1 Project Scope

The scope of this project is to provide an initial study of a robotic refueling system using a mobile base vehicle with a robotic arm attached. This thesis looks at the benefits of using a mobile-based vehicle for this task versus statically parking the base and performing a manipulation task from a single fixed point. Tests were conducted with a three-degree-of-freedom RE2 Automatic Arm to study the ability of a mobile manipulator to move along a desired trajectory to effectively align along the axis of insertion into a gas tank, and place a refueling tube securely in the tank to begin refueling. The primary goals for this study are to:

1. Investigate the feasibility for a three degree of freedom robotic arm to perform a refueling task at a given base location,

2. Among feasible trajectories, optimize the trajectory at a given base location,

3. Looking at different base locations, determine which spot is the best location for a refueling task, and determine if it makes sense to move, or park the mobile base,

4. Using results from goals 1, 2, and 3, identify the configuration requirements to build a mobile base for a robotic arm to optimize performance for a range of different fuel tanks.

The primary assumptions for this project are:

1. The refueling robot has successfully found and navigated up to the gas tank so the goal point is within the arm workspace,

2. The goal trajectory is known before carrying out the fueling task. In other words, the robot is not expected to learn how to insert a nozzle into each vehicles gas tank,

3. The refueling tube held by the end effector of the robotic arm will be mounted at a convenient orientation for insertion into the fuel filler tube,

4. The cap to the fuel filler tube has been removed, so the tube is open,

5. Because the focus of this thesis is on retrofitting a generic robotic manipulator arm for a refueling task, we are assuming the link lengths are a fixed design for this analysis,

6. The operation of the arm is done in a safe fashion such that refueling issues such as grounding, control of fumes around the equipment, safety around other humans, are already addressed independent of the motion control tasks considered here.

Chapter 4

Research Approach

This section provides an initial overview of the steps taken to address each goal indicated within the scope of the project. The refueling task has been broken into several sub analyses including:

1. Mathematically model a fixed-base and mobile base arm-configuration to determine appropriate joint angles and speeds to achieve a desired manipulator end tip location and velocity,

2. Simulate kinematic equations in MATLAB to validate arm end tip point P correctly follows a trajectory path,

3. Using an arm-only configuration, confirm the feasibility of successful insertion of the refueling tube into the gas tank. This is done by first building a CAD model of the arm and performing a refueling task in ROS, and then physically validating successful completion using the RE2 arm,

4. Investigate the arm behavior at different positions of a fixed base in relation to the gas tank, and compare the performance at each different base location,

5. Study the mixed motion of the robotic arm and mobile base in three-dimensions.

The following sections describe the process of addressing the kinematic and experimental analysis of the fixed and mobile base movement of the robotic arm.

Chapter 5

Mathematical Model of the RE^2 Robotic Arm and Mobile Base

5.1 Kinematic Model of Static Manipulator

An evaluation of the kinematics of the three-degree-of-freedom robotic arm was conducted using Denavit-Hartenberg parameters to describe the constraints of each joint. Denavit and Hartenberg parameters define the joint space of each revolute joint on the RE2 arm, defining the local coordinate frames for each joint and corresponding axis of rotation. These local joint coordinate frames are then described from one frame to another through transformation matrices. Figure 5.2 shows the coordinate system setup for each joint on the RE2 arm. Figure 5.1, taken from J. Craig's book, "Introduction to Robotics, Mechanics, and Control", illustrates these variables on a typical joint [1]. The Denavit-Hartemberg parameters for the RE2 arm are defined as follows:

a_i=Distance from Z_i to Z_{i+1} along X_i

α_i=Angle from Z_i to Z_{i+1} measured about X_i

d_i=Distance from X_{i-1} to X_i along Z_i

θ_i=Angle from X_{i-1} to X_i about Z_i

After defining the parameters from Table 5.1 for each joint, a transformation matrix can be obtained that provides a description of each joint in reference to the global base, or the zero frame. Further detail on this process can be found in

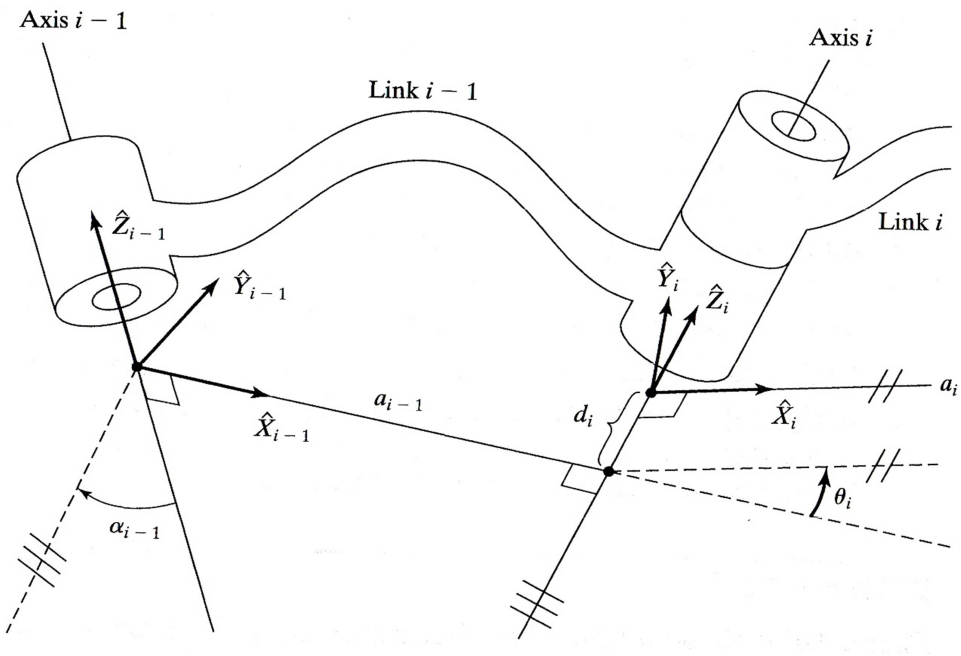

Figure 5.1. Illustration of Denavit-Hartenberg Joint Parameters [1].

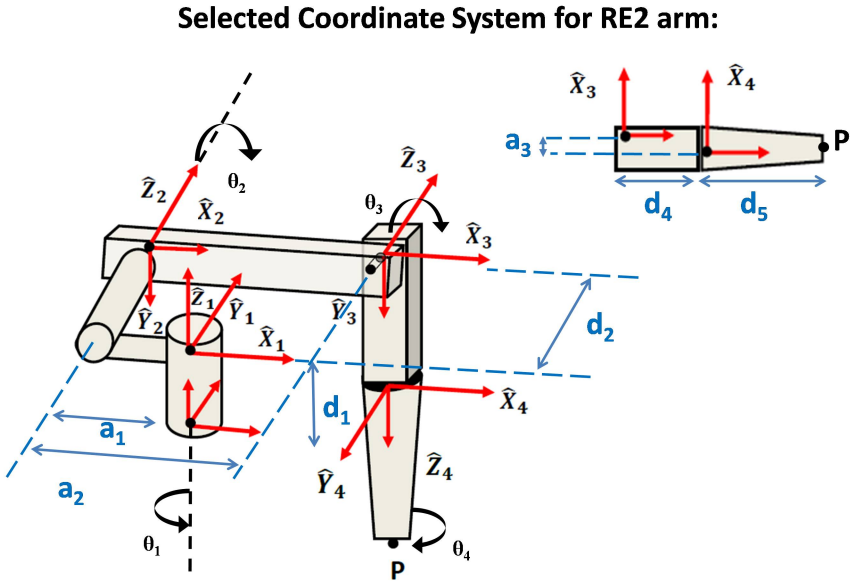

Figure 5.2. Diagram of RE2 arm with joint offsets.

Table 5.1. A list of Denavit-Hartenberg parameters used for a RE2 arm static base configuration for transformation matrices [1].

i	α_{i-1}	a_i	d_i	θ_i
1	0	0	d_1	θ_1
2	-90	a_1	d_2	θ_2
3	0	a_2	0	θ_3
4	-90	a_3	d_4	θ_4

several reference texts on robotics [1], [18]. A full description of all the derived transformation matrices can be found in Appendix section B.1. The notation in Equation 5.1 describes the transformation matrix from frame i-1 to i, using the appropriate parameters from frame i-1 and i illustrated in both Table 5.1, and Figure 5.2.

$$ {}^{i-1}_{i}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i\cos\alpha_{i-1} & \cos\theta_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1}d_i \\ \sin\theta_i\sin\alpha_{i-1} & \cos\theta_i\sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.1} $$

Using the transformation matrices, a description from the base frame to the end point P can be formed to describe the X, Y, Z location of point P in equation 5.2.

$$ \begin{bmatrix} P_X \\ P_Y \\ P_Z \end{bmatrix} = \begin{bmatrix} c\theta_1[a_1 + a_2c\theta_2 + a_3\cos\theta_{23} - s\theta_{23}d_5 - d_4s\theta_{23}] - d_2s\theta_1 \\ s\theta_1[a_1 + a_2c\theta_2 + a_3\cos\theta_{23} - d_4s\theta_{23} - s\theta_{23}d_5] + c\theta_1d_2 \\ d_1 - c\theta_{23}d_5 - d_4c\theta_{23} - a_3s\theta_{23} - a_2s\theta_2 \end{bmatrix} \tag{5.2} $$

Step 2 in Chapter 3 was conducted to verify the forward kinematic equations were correct for the RE2 arm. This validation process consisted of varying each joint from 0 to π or from 0 to -π. Using the forward kinematic equations, lines were plotted between the coordinates of each joint on the robotic arm. For this diagram, joint offsets a_1 and a_3 were set to zero. Joint offset d_2 was set to either 0.1 meters or 0 to facilitate the view of joint movements. Figures 5.3, 5.4, 5.5

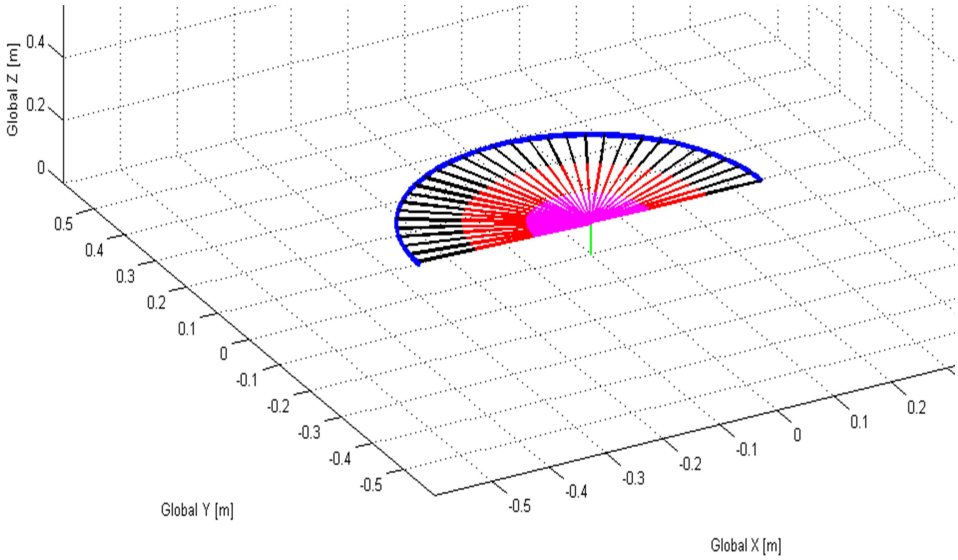

Figure 5.3. Static model of RE2 arm: θ_1 is varied from 0 to π; θ_2 and θ_4 are set to zero, and θ_3 is set to -90 degrees. Link 1 is colored green, link 2 is pink, link 3 is red, and link 4 is black. The end effector path is highlighted in blue. The first joint offset, d_2 is set to zero for better movement visualization.

illustrate the movement of the yaw joint, shoulder joint, and elbow joint from 0 to -π or π respectively. From these plots it can be seen that the joints will follow an arc, with the length of the arc being the sum of the linkage lengths.

5.2 Inverse Kinematics of Robotic Manipulator

Once the forward kinematic equations are validated, all joint angles can be found numerically using the Newton-Raphson iterative technique in equation 5.3. The Newton-Raphson technique is a quadratic convergence solution that arrives at a local minimum that is usually closest to the initial guess [18].

$$q^{(i+1)} = q^{(i)} + J^{-1}(q^{(i)})\delta T(q^{(i)}) \tag{5.3}$$

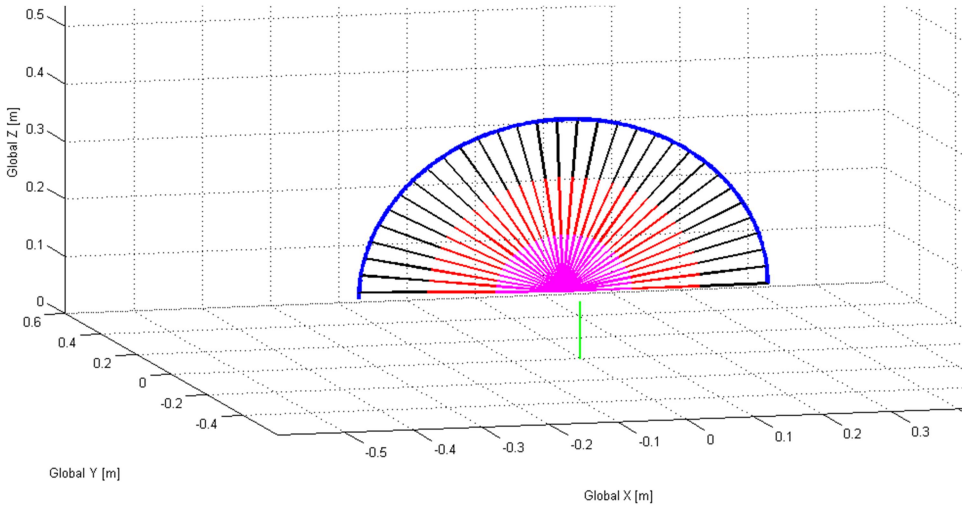

Figure 5.4. Static model of RE2 arm: θ_2 is varied from 0 to π, θ_1 and θ_4 are set to zero, and θ_3 is set to -90 degrees. Link 1 is colored green, link 2 is pink, link 3 is red, and link 4 is black. The end effector path is highlighted in blue.

Where,

$$q = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}, T = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, J(q) = \left[\frac{\partial T_i}{\partial q_j}\right] = \begin{bmatrix} \frac{\partial X}{\partial \theta_1} & \frac{\partial X}{\partial \theta_2} & \frac{\partial X}{\partial \theta_3} \\ \frac{\partial Y}{\partial \theta_1} & \frac{\partial Y}{\partial \theta_2} & \frac{\partial Y}{\partial \theta_3} \\ \frac{\partial Z}{\partial \theta_1} & \frac{\partial Z}{\partial \theta_2} & \frac{\partial Z}{\partial \theta_3} \end{bmatrix} \tag{5.4}$$

The Jacobian $J(q)$ of the arm is found by taking the partial derivatives of the expressions for P_X, P_Y, and P_Z with respect to θ_1, θ_2 and θ_3. An initial guess, q_i, must be provided to calculate the residual δT shown in 5.3. The residual represents the error between the location of the desired trajectory point, and where the arm is with present guess values. Using the Newton-Raphson equation in a looped MATLAB script, the equation will converge to an appropriate solution after a small number of iterations, stopping at a specified tolerance. As the end point of the manipulator moves along the trajectory path, the pervious pose of the arm will be used as a guess for the following trajectory point. The trajectory was

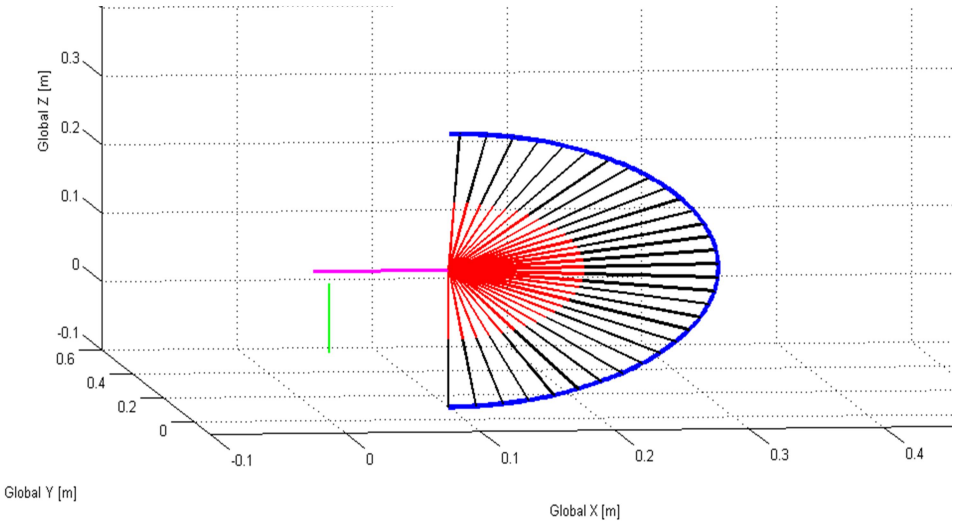

Figure 5.5. Static model of RE2 arm: θ_3 is varied from 0 to π, θ_1, θ_2, and θ_4 are set to zero. Link 1 is colored green, link 2 is pink, link 3 is red, and link 4 is black. The end effector path is highlighted in blue.

discretized into roughly 200 points per 9 inches, which is about 0.045 inch steps along the path. It is essential that the robot maintains an elbow-up configuration for an initial guess value; otherwise the iterative solver may converge to an elbow-down configuration. Once the appropriate Jacobian has been obtained with the correct joint velocities values from the Newton-Raphson solution, the joint angles of the system can be solved for using equation 5.5, where $\dot{q}$ represents the desired X, Y, and Z velocity values along the trajectory path.

$$\dot{\theta} = J^{-1}\dot{q} \tag{5.5}$$

5.3 Kinematic Model of Dynamic Mobile Base and Manipulator

The same process outlined for the static kinematic model of the manipulator was taken for the dynamic mobile base model. The model below illustrates mobile base movement through a prismatic joint. This constrains the mobile base movement along a single axis. This mimics the movement of a wheeled mobile base with no slip. Figure 5.6 illustrates the model and corresponding Denavit-Hartenberg parameters for the mobile manipulator. Table 5.2 shows the corresponding Denavit-Hartenberg parameters.

Table 5.2. A list of Denavit-Hartenberg parameters used for a RE2 arm mobile base configuration for transformation matrices [1].

i	α_{i-1}	a_i	d_i	θ_i
1	0	0	d_1	0
2	-90	0	d_2	θ_2
3	-90	a_2	d_3	θ_3
4	0	a_3	0	θ_4
5	-90	a_4	d_5	θ_5

The resulting expressions of the final transformation matrix describing the origin with respect to the end point P, and the X, Y, Z end point of point P with respect to all joint angles and offsets are described in equation 5.6.

$$\begin{aligned} P_X &= a_4 c\theta_2 c\theta_{34} + c\theta_2[a_2 + a_3 c\theta_3] - d_3 s\theta_2 - c\theta_2 d_5 s\theta_{34} - c\theta_2 d_6 s\theta_{34} \\ P_Y &= d_2 - d_5 c\theta_{34} - d_6 c\theta_{34} - a_3 s\theta_3 - a_4 s\theta_{34} \\ P_Z &= d_1 - d_3 c\theta_2 - s\theta_2[a_2 + a_3 c\theta_3] - a_4 c\theta_{34} s\theta_2 + d_5 s\theta_2 s\theta_{34} + d_6 s\theta_2 s\theta_{34} \end{aligned} \tag{5.6}$$

Because there are more variables in this system (d_1, θ_2, θ_3, θ_4, and θ_5) than there are constraint equations, the system will be underdetermined. This means that there are more columns in the Jacobian than rows.

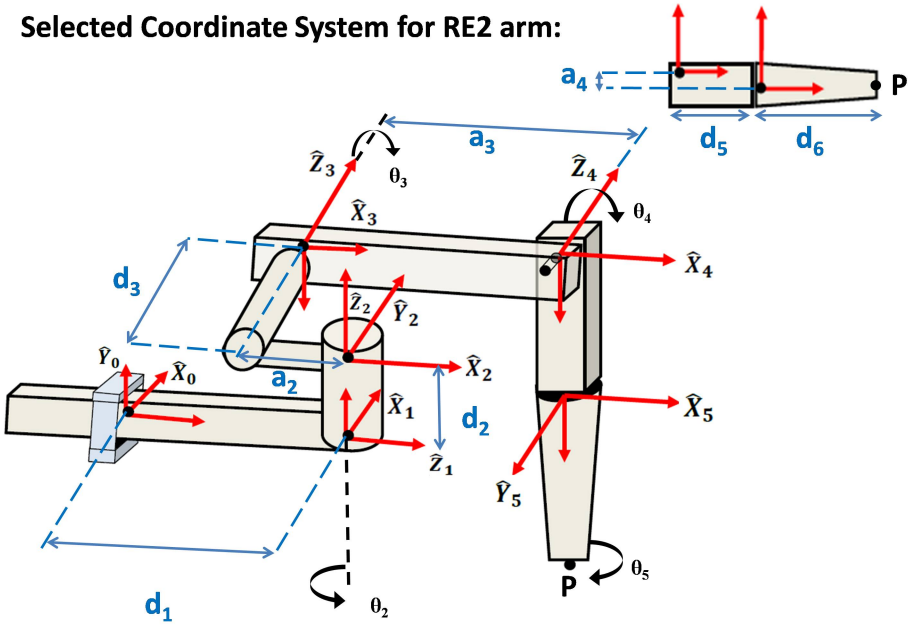

Figure 5.6. A model of coupled mobile base an manipulator with Denavit-Hartenberg parameters.

$$q = \begin{bmatrix} \theta_2 \\ \theta_3 \\ \theta_4 \\ d_1 \end{bmatrix}, T = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, J(q) = \left[\frac{\partial T_i}{\partial q_j}\right] = \begin{bmatrix} \frac{\partial X}{\partial \theta_2} & \frac{\partial X}{\partial \theta_3} & \frac{\partial X}{\partial \theta_4} & \frac{\partial X}{\partial d_1} \\ \frac{\partial Y}{\partial \theta_2} & \frac{\partial Y}{\partial \theta_3} & \frac{\partial Y}{\partial \theta_4} & \frac{\partial Y}{\partial d_1} \\ \frac{\partial Z}{\partial \theta_2} & \frac{\partial Z}{\partial \theta_3} & \frac{\partial Z}{\partial \theta_4} & \frac{\partial Z}{\partial d_1} \end{bmatrix} \tag{5.7}$$

Because of this, an additional constraint, or driver equation must be defined in order to solve the system. To solve this problem, kinematic equations for the arm-only configuration shown in equations 5.2, 5.3, and 5.4. Arm performance metrics, such as maximum joint velocities, were then evaluated based upon the static model to select a specific base location with a d_1 offset for a given trajectory. This chosen d_1 value can then be substituted into the mobile base model. Further discussion of this process is given in Chapter 10.

Chapter 6

Optimal Base Selection

When the mobile robotic arm receives a command to perform a trajectory task, the robot must first locate obstacles in its surrounding reachable workspace. This obstacle data can be detected via stereoscopic cameras, LIDAR, or other forms of sensing that will return 3D point cloud data, triangulated into established obstacle faces and vertices. As outlined in section 3.1, it is assumed that an optimal end effector trajectory path will be pre-programmed into the robot for the fuel tank to be filled. The concern of this thesis is to evaluate which trajectory and mobile base location will best position the robot relative to obstacles and the desired nozzle path. In the field, a manipulator can shift or slip while performing a refueling task due to uneven or loose terrain. Taking this into account, the robot must establish where to position the base so that, if these errors do occur, the robot is still positioned to successfully complete a refueling task. Thus, the existence of a feasible path is not sufficient for the robot; a good trajectory must be nearby many feasible paths that are reachable to each other without singularities, object collisions, saturations in joint velocities, etc.

Figure 6.1 shows a high level functional diagram of the robots analysis of the surrounding workspace and execution of a refueling task. This section is concerned with the function diagram block in Figure 6.1 titled "Pick Optimal Base Location".

To determine an optimal base location, a brute-force approach is used to analyze the outcome of each refueling task along a grid of possible base locations relative to the trajectory path. Figure 6.2 illustrates a grid of base locations to be analyzed relative to the trajectory path. The first step of this analysis is to map

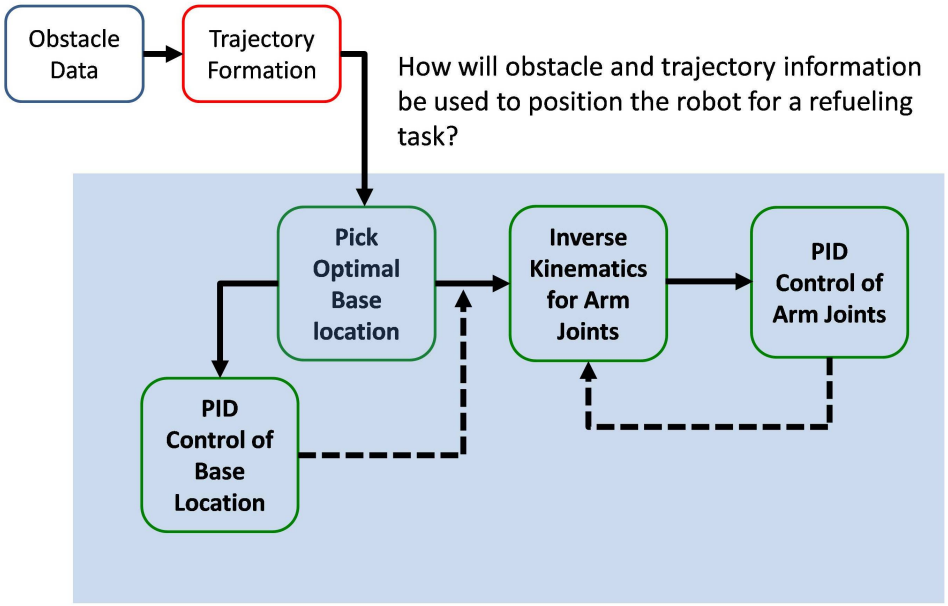

Figure 6.1. High-level function diagram of a robotic manipulator performing a vehicle refueling task. The focus of this study is highlighted in blue.

out joint singularities, joint limit violations, and any collisions that may occur at a specific base location.Once these violations have been recorded, both velocity and mass data are used to evaluate a series of cost functions. Each base location is then assigned the resulting cost value, and the 2D grid of costs are then plotted to create a 3D cost map. This 3D cost map serves as a metric to illustrate the difficulty in completing a refueling task from a specific base location. The details of this analysis are explained in the remaining sections below, beginning with the identification of joint limit violations that may occur during a refueling task.

6.1 Reference Trajectory

All 3D cost plots will be for a final reference trajectory representing nozzle insertion to maintain consistency throughout the thesis. This will also allow direct comparisons to be made between all cost functions. The reference trajectory will be a nine inch straight horizontal path followed by a 90 degree vertical turn downwards for four inches. Other trajectories can be considered where the inlet may be at a 65 or 45 degree angle from the horizontal. The end effector tip velocity will travel along

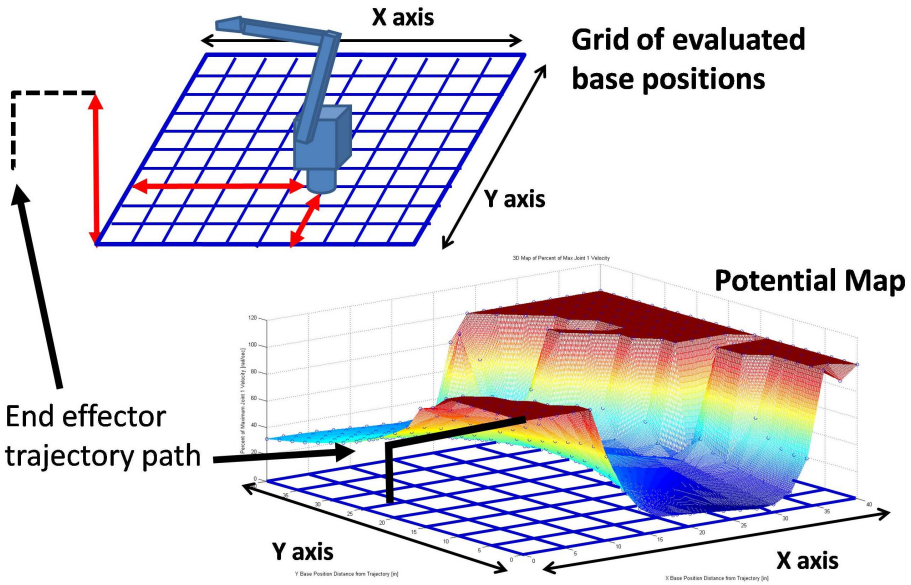

Figure 6.2. Diagram illustrating grid layout used to analyze performance of refueling task at different base positions.

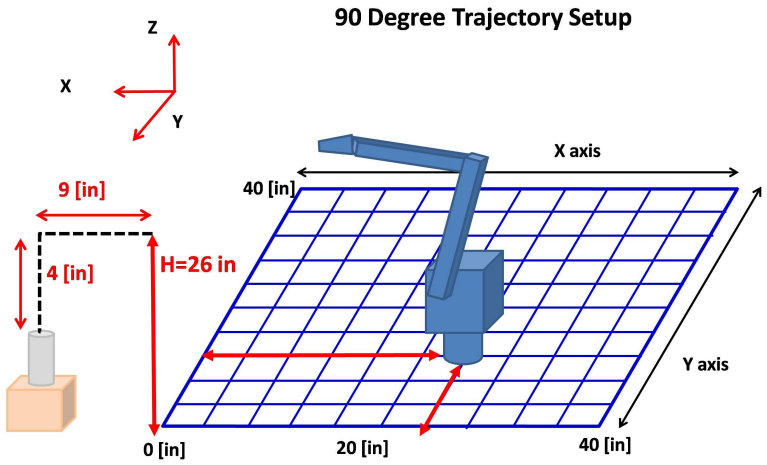

Figure 6.3. Picture of trajectory setup used for plots below. The base location is a square grid, 40 in X 40 in, with increments of 2 in varied for the (X, Y) base location of the arm. The end tip must travel at 0.1 m/s in the x direction, and -0.1 m/s in the z direction at the 90 degree portion of the path.

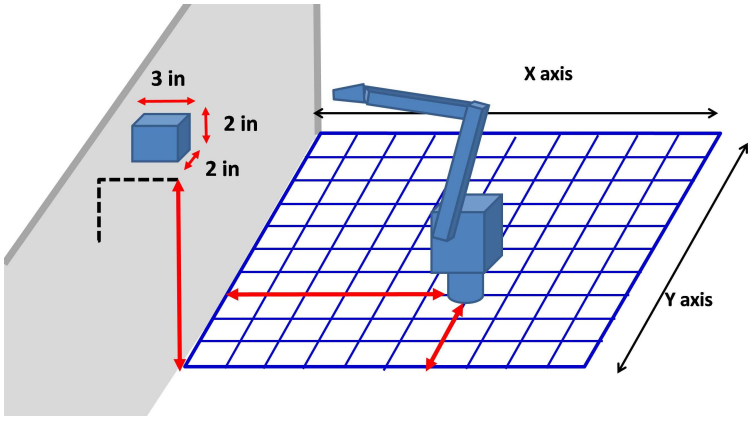

Figure 6.4. Illustration of obstacle setup for collision testing. This obstacle mimics extrusions located near the gas tank of various vehicles.

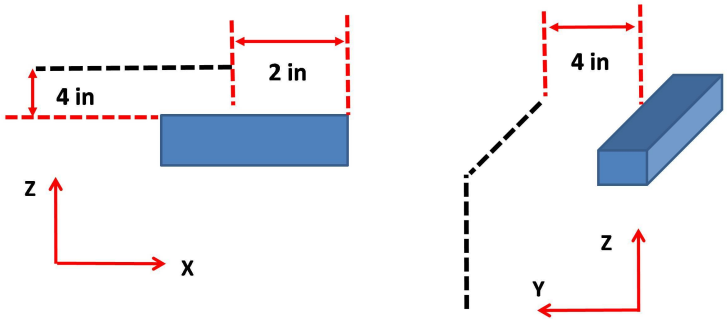

Figure 6.5. Diagram illustrating layout of simulated obstacle in relation to the end effector path.

the x-axis at 0.1 m/s, and for the latter part of the trajectory, it will travel at -0.1 m/s along the z-axis. For this specific case, a sharp change in the trajectory path will cause an abrupt change in velocity at the corner of the path. In the field, it is suggested that the pre-programmed path smoothes all these corners to eliminate sharp changes in the end effector tip velocity, and corresponding joint velocities. A sample view of this trajectory is shown in Figure 6.3.

In addition to the trajectory path, an obstacle was also placed near the desired trajectory path to see how collision information affected the available base locations for the mobile robot. There are many cases where obstacles are located near the gas tank of the vehicle that the robot would need to avoid. An example of this can be found when looking at Figure 6.4, of a sample refueling tank, where obstacles

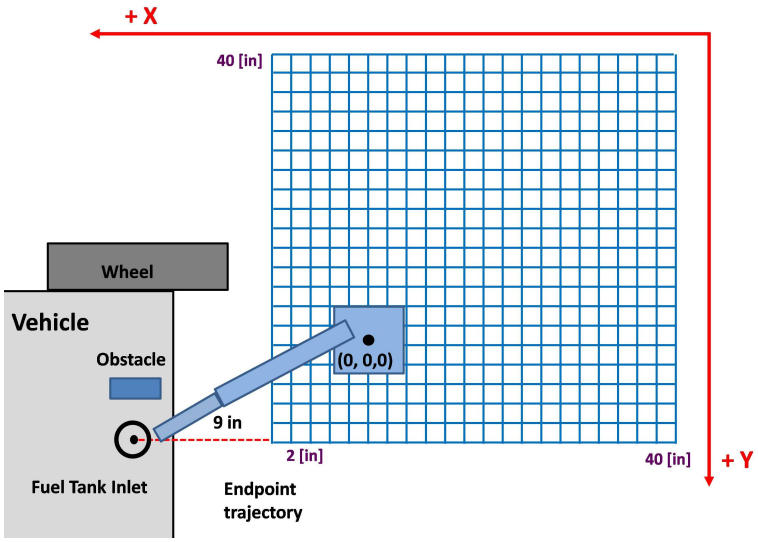

Figure 6.6. Aerial view of base location grid, sample end effector trajectory and sample fuel tank inlet layout. The origin of the robotic manipulator serves as the global origin.

are found to the right of the fuel inlet. The obstacle was generalized as a box located near the trajectory path. More details on the obstacle dimensions are found in Figure 6.5, and an aerial view of the trajectory layout in relation to the fuel tank is found in Figure 6.6.

6.2 Cost Functions

With this study, a series of individual cost functions were considered based on the following goals:

- Map out joint limit violations
- Identify joint singularities
- Map out collision occurrences along trajectory path
- Minimize maximum joint velocities along trajectory path
- Minimize shoulder, or other joint movement
- Minimize power consumption

Each of these cost functions will be described in the following sections throughout this chapter. All of these cost functions will then be applied using the reference trajectory illustrated in section 6.1.

6.2.1 Determining Joint Limit Violations

Every robotic manipulator has specific joint limit constraints that are based upon the design of the robot. There are constraints on the range of motion of the motors themselves, and also includes constraints on elbow down configurations that could cause the manipulator to collide with itself or the ground below it. All joint limits discussed below are specific to the RE2 arm linkage dimensions.

The first joint-limit constraint for the RE2 arm is due to limits on where the second link of the arm, the shoulder joint, can move. As seen in Figure 6.7, there is a tooling stand that the RE2 arm can collide with at -166 degrees from the reference zero point. Because of this tooling stand, a constraint on the shoulder arm has been implemented to flag a violation if the arm is outside of the boundaries of -166 $< \theta_2 <$ -14 degrees. This limit is also useful to prevent elbow-down configurations that could cause the elbow joint to collide with the mobile base. It is important to note that the joint angle calculations are consistent with the Denavit-Hartenberg coordinate frames outlined in Chapter 5.

The elbow joint is also flagged for joint limit violations. The elbow joint configuration only has limitations when the elbow joint causes links 2 and 3 to fold in on link 1. This limit occurs at joint values consistent with the Denavit-Hartenberg coordinate frame at $80 < \theta_3 < 100$ degrees. Figure 6.8 below illustrates this joint limit.

While calculating the appropriate joint angles with the Newton-Raphson iterative calculator, sometimes the solution will converge to an angle value that is a multiple of π from the actual angle. Because theses angles are sent to the RE2 arm OCU, all joint angles are unwrapped using a MATLAB function, causing joints to range from -180 to 180 degrees. This unwrap function is referenced in Appendix section B.1. The method of flagging joint limits works in this simple example; however, a more robust method consists of performing a collision detection calculation. A collision detection algorithm allows the arm to avoid more complex

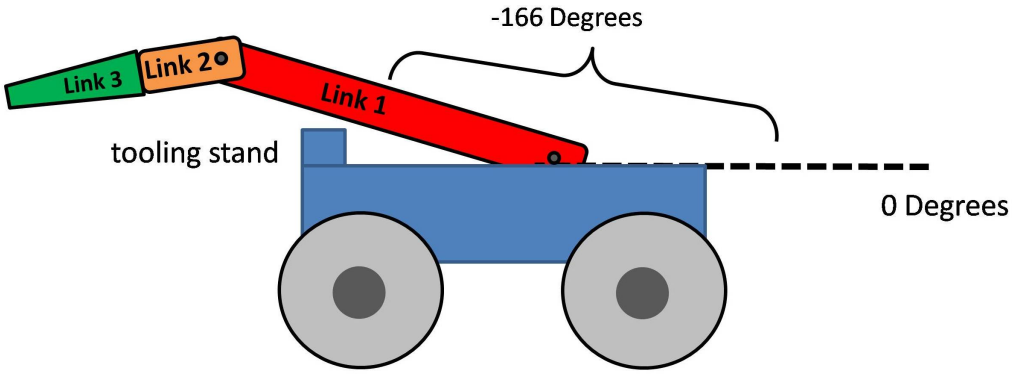

Figure 6.7. Illustration of shoulder joint limit constraints. The shoulder joint is limited by boundaries from -166 $< \theta_2 <$ -14 degrees.

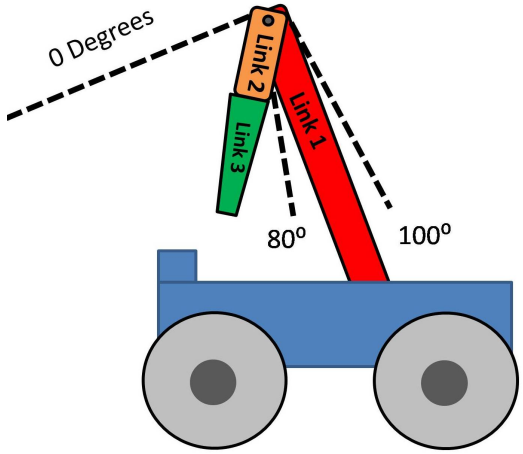

Figure 6.8. Diagram illustrating elbow joint limits on manipulator from 80 to 100 degrees.

extrusions from the base rather than just simple geometrical constraints on the linkages. However, this would require a time-consuming ray-casting (or similar) algorithm, and for this thesis, such error checking is unnecessary given the simplicity of the robots design. Figures 6.9 and 6.10 are pictures illustrating joint limit violations that occur at certain base positions on the grid for the given reference trajectory.

It can be seen in Figures 6.9, 6.10, that joint violations occur at the outer

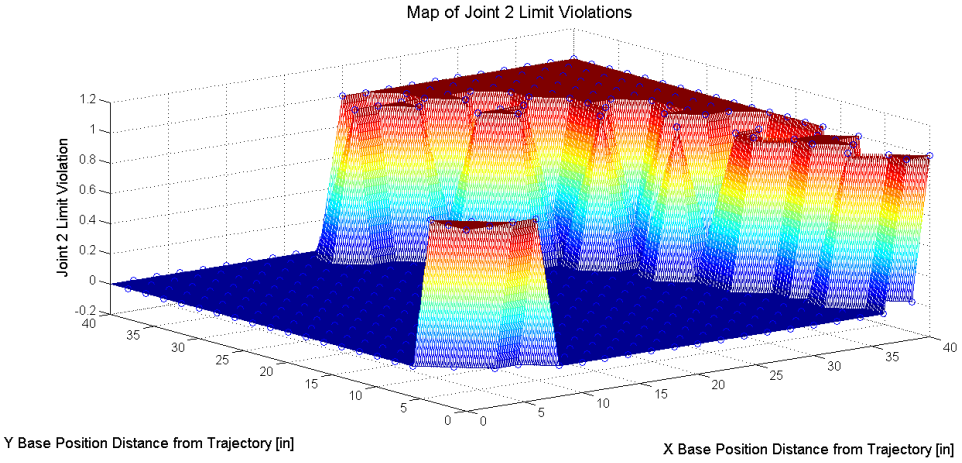

Figure 6.9. Cost map plotting joint 2 violations that occur at each (X, Y) base position. A violation occurrence is labeled as either true (1) or false (0), this cost map is for a reference trajectory at a height of 26 inches.

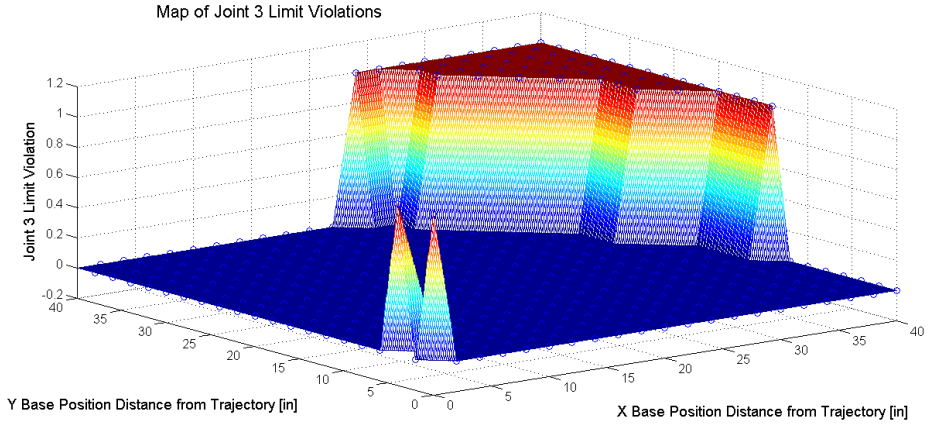

Figure 6.10. Potential map plotting joint 3 violations that occur at each (X, Y) base position. A violation occurrence is labeled as either true (1) or false (0), this cost map is for a reference trajectory at a height of 26 inches.

edges of the grid where the mobile base is furthest from the desired trajectory. These violations are occurring because the further points along the trajectory lead to joint singularities, due to the inability of the arm to stretch further than full extension of individual joints. When the Newton-Raphson solver reaches these points of singularity, the solution will diverge. These results are reflected in the furthest edges of the plots above. Because the base is so far away from the refueling location on this perimeter, many of the trajectory points are on the limits or even

outside of the RE2 arm's workspace. These results illustrate that it is important to map out the workspace of the robotic arm to understand how these violations coincide with joint singularities.

6.2.2 Workspace of the Arm

The workspace of the arm is a map of all the reachable points of the robotic arm, taking into account all of the joint limit constraints. The workspace of the arm will consist of all reachable points with the arm outstretched. Normally, the arm would span a half circle dome of reachable space, with a dome radius equal to the length of the extended linkages. However, because there are joint limitations on where the shoulder joint can be in relation to the tooling stand, the arm will only extend out to 14 degrees from the horizontal. This is illustrated in Figure 6.11.

Any trajectory points that lie on the outer edge or outside of the highlighted workspace will lead to a diverging solution using the Newton-Raphson iterative calculator. This will then show up as a violation for the corresponding base position on the cost map. Only the areas within the highlighted blue section are reachable by the manipulator arm. The outer singularities can be found simply by calculating the distance from the trajectory point to the coordinate of the shoulder joint. If the distance between the two coordinates is greater than or equal to the length of the links, there will be no possible solution and the Newton-Raphson calculator will diverge. Equation 6.1 illustrates the distance calculation for the two coordinates. Variables for the linkage lengths are found in Table 5.1, Chapter 5. A plot of singularities is shown in Figure 6.12 for the reference trajectory of height 26 inches.

$$\begin{aligned} d_{trajectory} &= \sqrt{(X_{trajectory} - X_2)^2 + (Y_{trajectory} - Y_2)^2 + (Z_{trajectory} - Z_2)^2} \\ d_{trajectory} &< a_2 + d_4 + d_5 \end{aligned} \tag{6.1}$$

6.2.3 Collision Detection

Obstacles near the desired trajectory path may limit the locations where the mobile robot can navigate to. In some configurations, links can collide with obstacles while

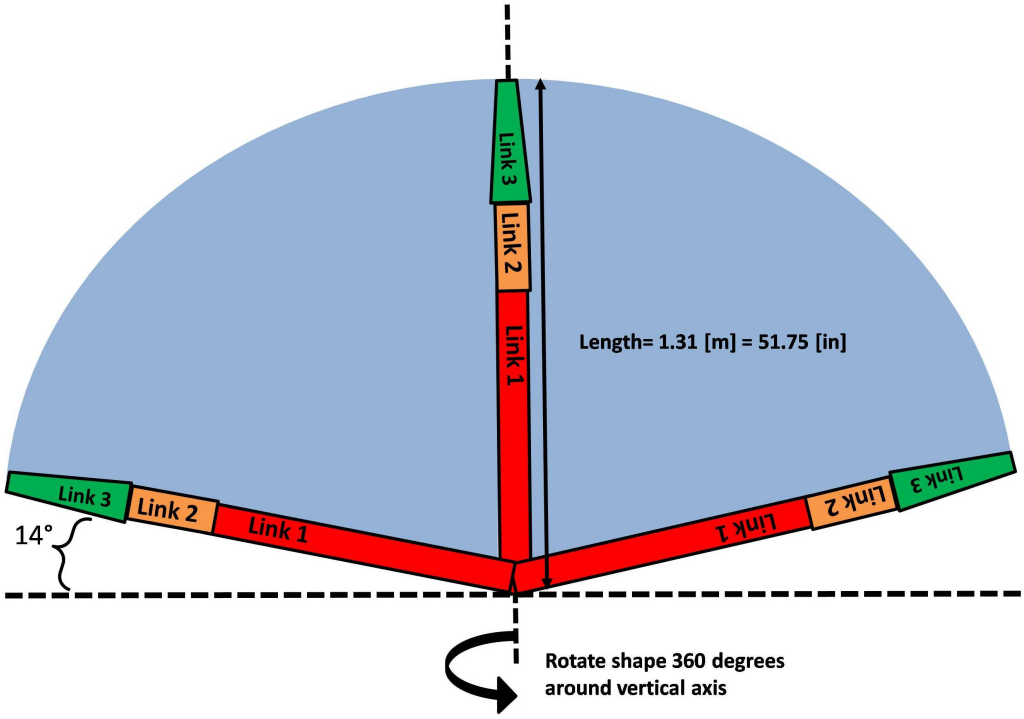

Figure 6.11. Cross-section illustration of RE2 arm 3D reachable workspace.

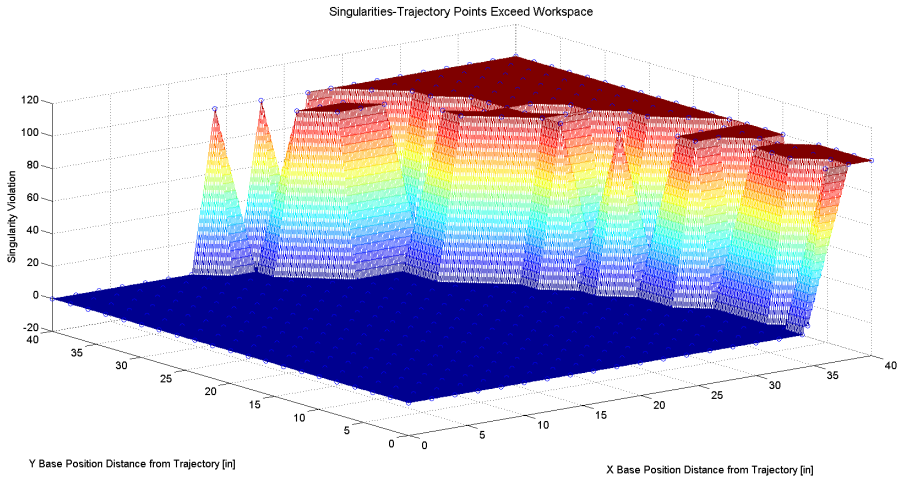

Figure 6.12. Plot of singularities that occur when a trajectory point lies outside of the robotic arms workspace.

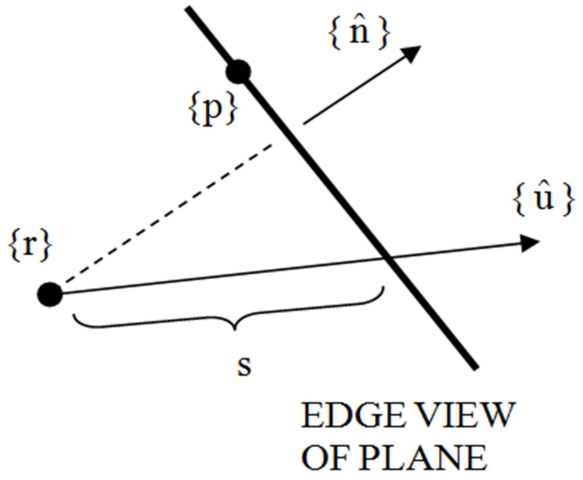

Figure 6.13. Diagram of intersection of ray and plane [2].

following a trajectory path and these base locations must be marked as a collision violation.

The method used to detect obstacle collisions is called ray casting. For this implementation, a MATLAB script checks for the intersection of a line and a plane. The line represents the linkages for the arm itself, and the plane represents an obstacle face near the trajectory path. The process of ray casing consists of creating an infinite line that is in the direction of the line that you are checking. Using the infinite line, the ray casting method calculates the distance between point r and the point of intersection along the infinite line. If there is no intersection point, the line is parallel to the obstacle face. Figure 6.13 illustrates the intersection of a line and a plane, with the intersection distance s. Equation 6.2 is the formula to calculate distance s from Figure 6.13, describing the intersection of a line and plane [2]. Given:

r = known location on ray

$\hat{u}$ = known unit direction of ray

p = known location on plane

$\hat{n}$ = known unit normal to plane

$$s = \hat{n}^T(p - r)/(\hat{n}^T\hat{u}) \tag{6.2}$$

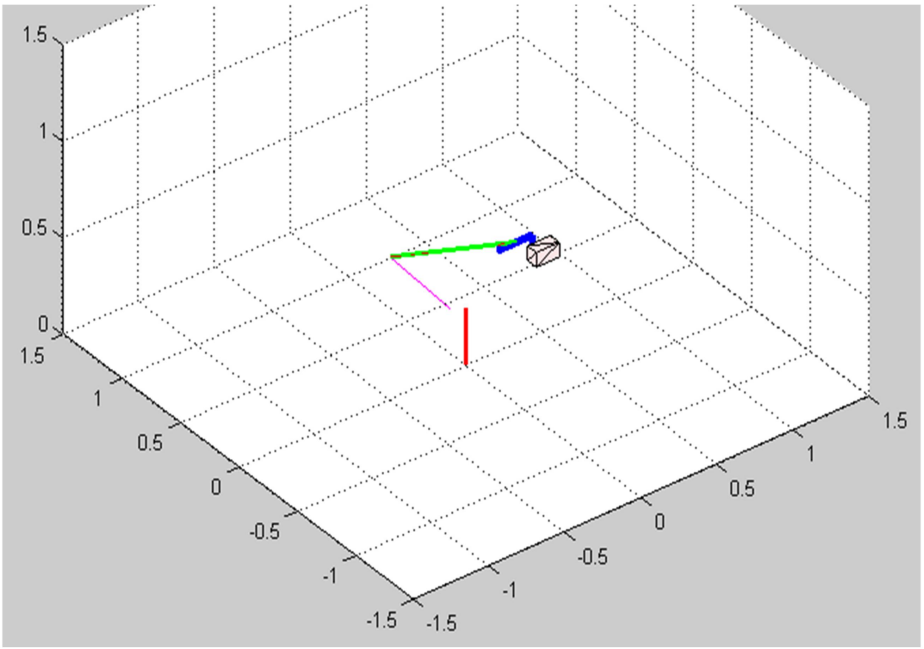

Figure 6.14. Picture of RE2 arm moving along a trajectory in a MATLAB simulation. Links 2 and 3 are being checked for collisions with obstacle seen to the right of the trajectory path. All axis units are in meters.

Once the value of s is computed, one can check if the length of the s is greater or less than the length of the linkages. If distance s is less than or equal to the length of the linkage, then there is a collision between the links and the planar obstacle [2]. This process is repeated to check all faces of the obstacle. For the RE2 arm application, the last two links were the only linkages checked for collisions. This was done because it was already known that the shoulder link did not intersect with the obstacle with the current arm geometry. However, in unknown environments, all arm linkages would need to be checked. Figure 6.14 is a picture of a ray casting check within MATLAB. The picture shows the lines of the RE2 linkages as it moves along the trajectory. The obstacle seen near the trajectory was created in SolidWorks. The model was saved as an STL file in binary format to be imported into MATLAB. The ray casting method was used to check all faces of the imported SolidWorks file for collisions.

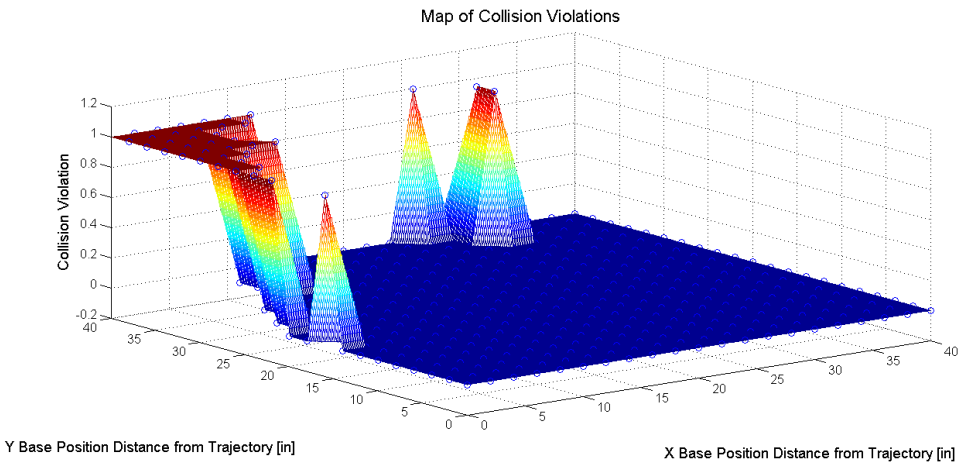

Figure 6.15. Plot of collision violations for each base location.

Because this ray casting method approximated the arm linkages as lines, the obstacle can be made much bigger than it actually is to compensate for the actual geometry of the robotic arm linkages. This process is called dilation [19]. The dilation process is useful in speeding up computation times when performing the collision check algorithm within MATLAB.

Figure 6.15 is a picture of base locations where a collision occurs while traveling along the trajectory at a certain height, seen in Figure 6.4. The obstacle prevents the mobile manipulator from successfully carrying out a refueling task at the back left locations on the grid. These base positions will cause links 2 and 3 of the manipulator to collide with the obstacle towards the end of the followed trajectory path. A detailed MATLAB script of the ray casting calculation can be found in Appendix section A.2.

6.2.4 Percent of Maximum Joint Velocity

After collision and joint limit violation locations are evaluated, velocity solutions for each joint were checked for each base location as the manipulator follows the specified trajectory path. In order to find the percent of maximum joint velocity commanded on all three joints, the maximum velocity achievable from the RE2 arm was measured. Each motor on the RE2 arm was commanded to travel 180 degrees at 100% effort. The time taken to complete this task was recorded. The average velocities for each motor are shown in Table 6.1.

Table 6.1. A list of approximate average maximum yaw, shoulder and elbow joint velocities.

Yaw joint θ_1	Shoulder joint θ_2	Elbow joint θ_3
16 deg/sec	17 deg/sec	21 deg/sec

$$\%\dot{\theta}_{1,Max} = \frac{\dot{\theta}_1}{\dot{\theta}_{1,Max}}, \%\dot{\theta}_{2,Max} = \frac{\dot{\theta}_2}{\dot{\theta}_{2,Max}}, \%\dot{\theta}_{3,Max} = \frac{\dot{\theta}_3}{\dot{\theta}_{3,Max}} \tag{6.3}$$

Yaw joint measurements were taken with arm pointing straight up, and measurements for the shoulder joint were taken with links 1, 2 and 3 aligned. It is logical that the elbow joint would move much faster than the shoulder and yaw joint as there is less mass for the elbow joint to move, in comparison to the shoulder and yaw movement of the whole arm. For the percent of maximum cost plots in this section, maximum joint velocity values assigned to each joint were: Yaw Joint=19 deg/sec, Shoulder Joint=18 deg/sec, and Elbow Joint=25 deg/sec. Once the maximum joint velocities were assigned, each resulting joint velocity found from the Newton-Raphson calculator for each trajectory point was divided by the maximum joint velocity to find the percent of maximum velocity for each joint, as shown in Equation 6.3.

A cost function based on the percent of maximum velocity is useful in comparing different robots, particularly ones with similar arm geometries but different motors. Specifically, one would expect that the same arm using motors with different performance characteristics would give different cost functions, thus allowing one to design the motors for that robot arm to best achieve a specific task such as refueling.

Using the percent maximum joint velocity as a cost function, a 3D plot was created showing the maximum percent joint velocities for all possible base locations. Each joint was looked at individually, and a corresponding cost plot was created that extracted the maximum percent joint velocity for the entire trajectory task. This maximum percent joint velocity across all joints was then assigned as the cost associated with the corresponding base location.

As an example, assume that the mobile base is located on the (X, Y) grid at a location of (20 in, 20 in), and the arm end point P is commanded to follow a

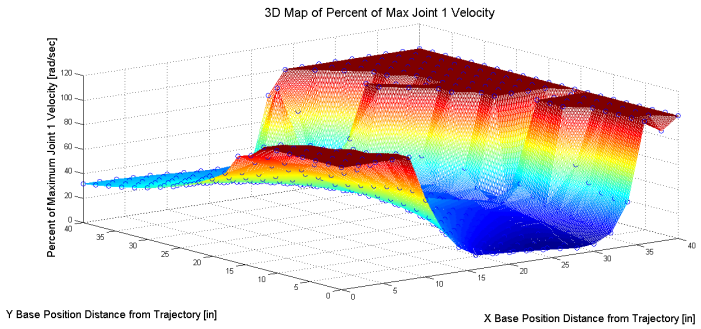

Figure 6.16. Plot of maximum joint 1 velocity for 90 degree trajectory at each base location.

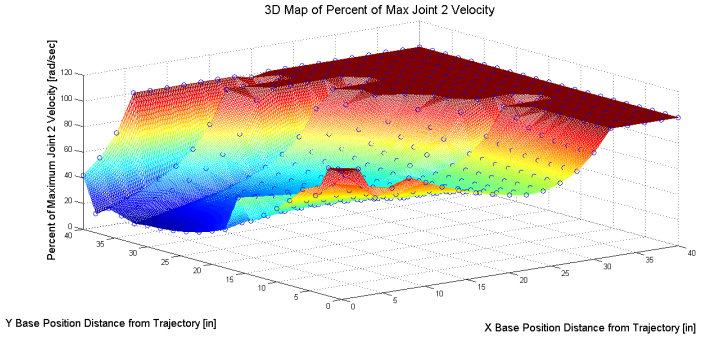

Figure 6.17. Plot of maximum joint 2 velocity for 90 degree trajectory at each base location.

fixed trajectory path. The trajectory itself is split into 250 individual trajectory points. For each trajectory point, there are three joint angles (θ_1, θ_2, θ_3) and three joint velocities that enforce the requirement that the arms end-point P will follow the specific trajectory point at the desired velocity. For an entire trajectory, there will be 250 values of θ_1, and 250 values of $\dot{\theta_1}$. One can examine all joint 1 velocity values to find the maximum velocity value for the entire trajectory. The percent maximum velocity cost function for that base location, which is (20, 20) is then assigned the corresponding maximum θ_1 velocity value. Any values exceeding 100% are cut off at 100%, as this is a violation of the maximum possible velocity. Cost maps for the yaw, shoulder, and elbow joint are shown in Figures 6.16, 6.17, and 6.18.

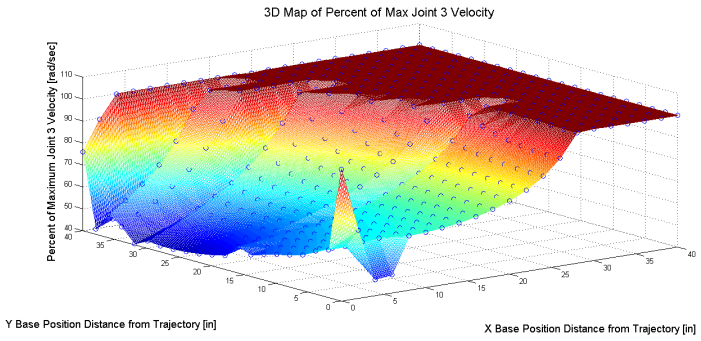

Figure 6.18. Plot of maximum joint 3 velocity for 90 degree trajectory at each base location.

6.2.5 Cost Function 1

By taking the worst-case costs at each base location normalized to 0 and 100 for each cost function, a complete cost function was next created that combined collision, joint limit, and velocity data into a overall cost function for the mobile robot.

The first cost function considers the maximum joint velocity that occurs across the 3 joint angles. This cost function is the same as simply plotting out the maximum velocity for each base position for one joint, except this method takes the maximum value across all three joints. The MATLAB script follows the format below, where all matrices are of size [250X1].

Maximum Velocity =max(θ_1, θ_2, θ_3)

Cost Value = max(Maximum Velocity, Limit 2 Violation, Limit 3 Violation, Collision Violation)

Once a maximum velocity value is extracted from this list for each base location, these values must also be compared against joint limit, and collision violation values. By doing this check, all maximum velocity values for a base location are then overridden by collision or joint violations that may occur at that location. This process thereby combines all violation and velocity data into one plot. This cost function that incorporates all velocity, collision, singularities, and joint limit costs is shown in Figure 6.19.

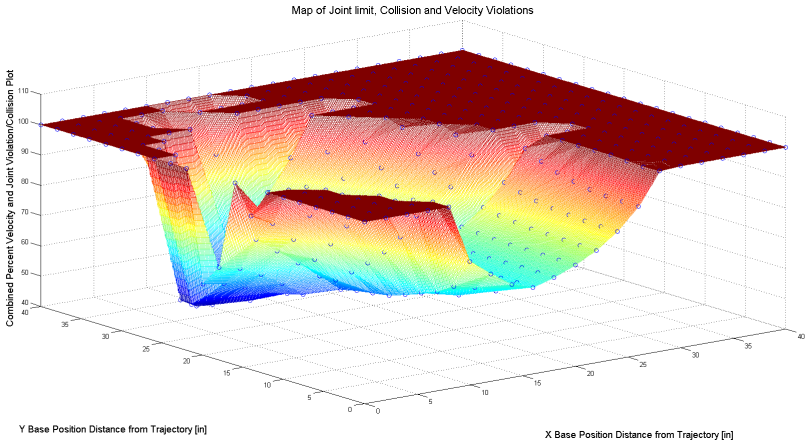

Figure 6.19. Cost function including velocity, collision, singularity, and joint violation costs combined. The cost function extracts maximum joint velocity for each base location across all three joints. This map considers the 90 degree trajectory at a height of 26 inches.

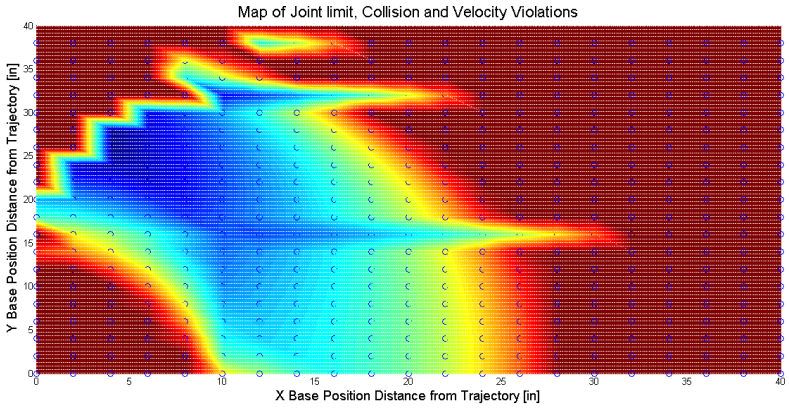

Figure 6.20. Aerial view of Figure 6.19 of the combined cost function. This view illustrates the base locations with low costs, where minimal velocities occur in blue and high velocities, collisions, singularities, and limit violations occur in red.

6.2.6 Cost Function 1 for Individual Trajectory Points

All cost function plots shown in the previous sections in this chapter illustrate the worst case velocity, collision, or joint limit violation value that occurs throughout the entire trajectory path. This is done to ensure that the mobile robot avoids base positions that could lead to a singularity, collision, or joint limit violation that could occur while it follows the trajectory path. This does not, however, mean that every single trajectory point has a joint limit or collision violation. In

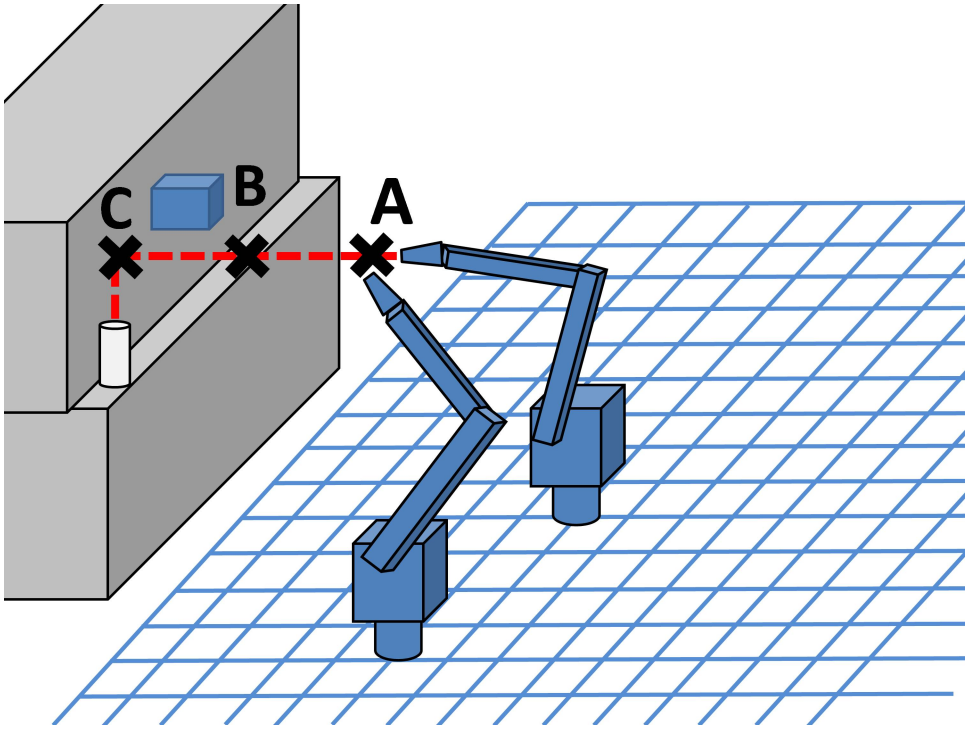

Figure 6.21. Illustration of sample trajectory points A, B, and C along the trajectory path. The mobile robot can reach these points from different base locations.

this section, individual trajectory points will be assessed to illustrate how the cost map changes for each individual trajectory point. This is done to illustrate why it may be beneficial for the robot to move the base to a different position as the end effector travels from one coordinate point to the next along a desired path. For this example, three coordinates have been chosen to illustrate the difference in cost plots along the trajectory. An illustration of the trajectory points chosen are shown in Figure 6.21.

The location of points A, B, and C are:

A=5^{th} trajectory point, 0.25 (1/4) inch from the initial start point along the 9 inch horizontal portion of the trajectory.

B=85^{th} trajectory point, 3.8 inches from the initial start point the 9 inch horizontal portion of the trajectory.

C=200^{th} trajectory point at 9 inches from the beginning of the horizontal portion of the trajectory. Point C is the last point before the trajectory moves vertically down 4 inches.

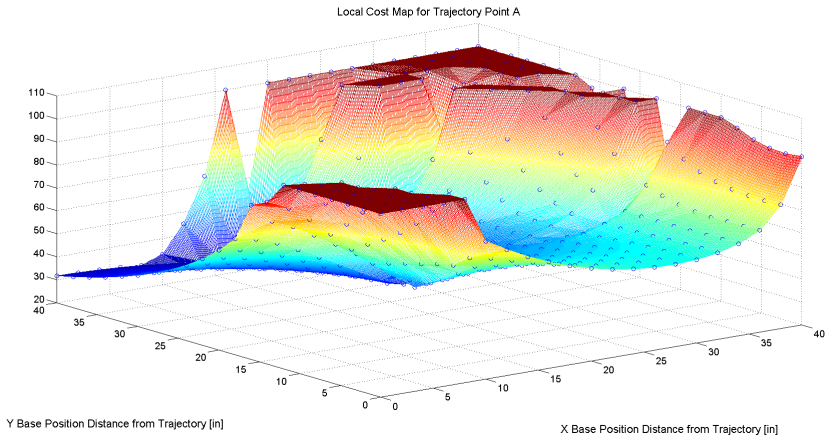

Figure 6.22. Combined collision, singular, joint limit, and velocity data for point A.

Cost functions were created for points A, B, and C that mapped out all collisions, joint limit violations, singularities and velocity limits that occurred for each the base position. This data was combined with the velocity costs to provide a final overall cost for each base location, following the same process as the previous section for cost function 1. The only difference between this section and the last is this cost plot is for a single point, rather than pulling out the worst cost along the entire trajectory. Because each trajectory point is located at a different distance away from the origin of the arm, points further away, such as C, and B, will be more restrictive in where the base can navigate to in comparison to the closer point A. Cost plots for points A, B, and C are shown in Figures 6.22 through 6.27. Looking at point C, it can be seen that it would be beneficial for the robotic arm to drive the base forward to move closer to the trajectory points that occur further away from the arm, to maintain optimal performance metrics, rather than just keeping all points along the trajectory within the workspace.

6.2.7 Cost Function 2

In addition to minimizing joint velocities and avoiding any violations that may occur during the refueling task, it is also useful to explore other cost functions that can minimize specific joint movements. This can allow the robot to take into account the performance of certain joint motors before it carries out a refueling task. For example, if the shoulder joint on the robotic manipulator is damaged

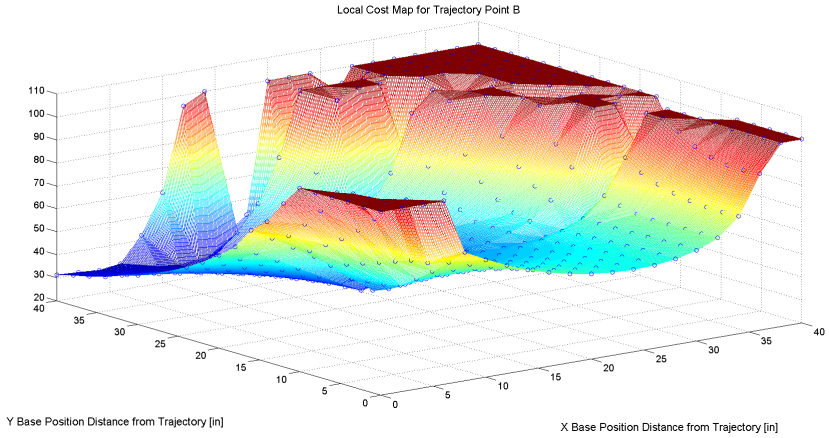

Figure 6.23. Combined collision, singular, joint limit, and velocity data for point B.

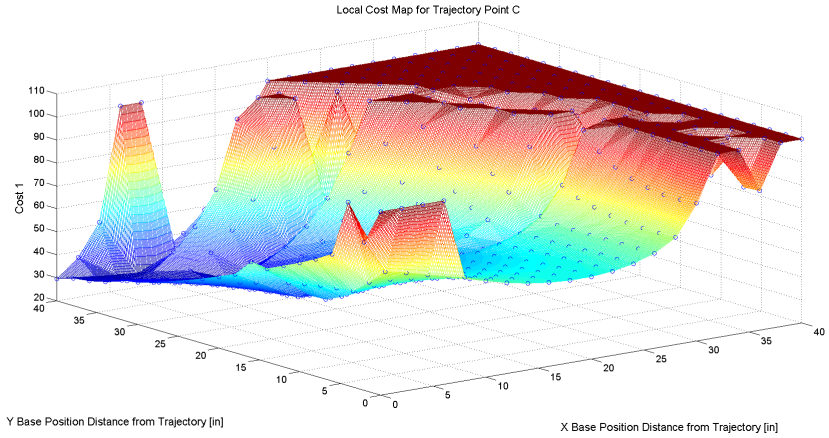

Figure 6.24. Combined collision, singular, joint limit, and velocity data for point C.

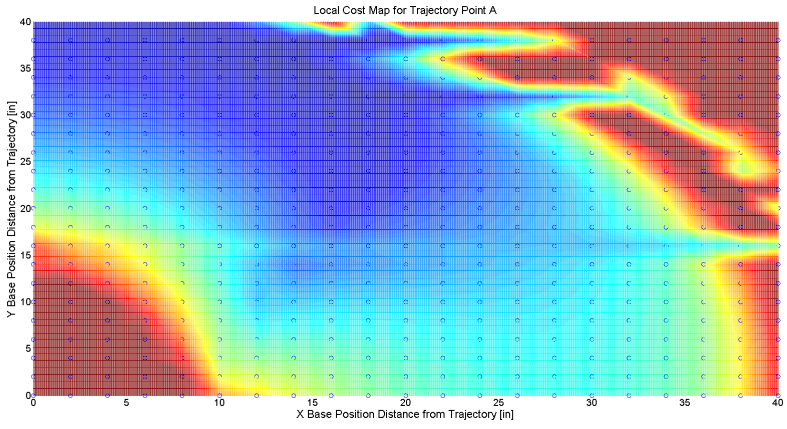

Figure 6.25. Aerial plot of cost map from Figure 6.22 for point A.

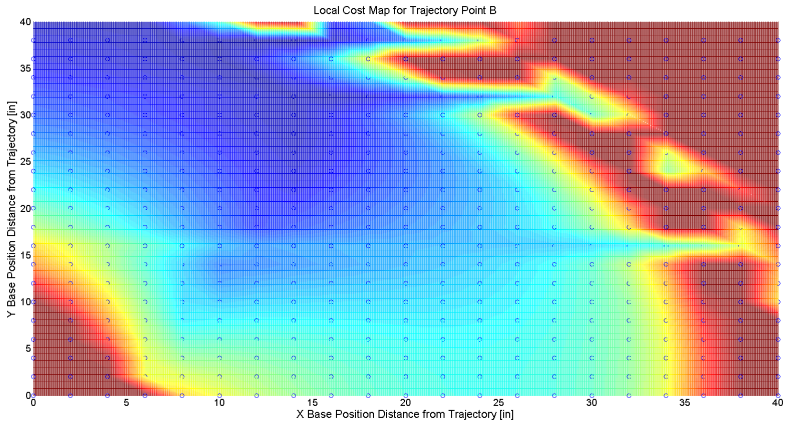

Figure 6.26. Aerial plot of cost map from Figure 6.23 for point B.

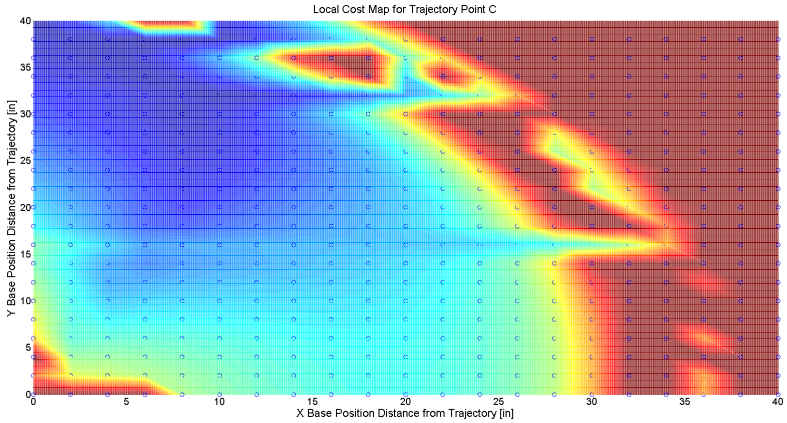

Figure 6.27. Aerial plot of cost map from Figure 6.24 for point C.

and running at a fraction of its normal performance, a cost function can be used to penalize shoulder movement during a refueling task. For this example, a weighted cost penalizes the shoulder velocity 3 times as much as the yaw or elbow joint. The total weighted cost is shown in equation 6.4.

$$Cost_2 = 0.2 \times \dot{\theta}_1 + 0.6 \times \dot{\theta}_2 + 0.2 \times \dot{\theta}_3 \tag{6.4}$$

Once a total cost is established for each base location, these cost values are then combined with the joint limit, and collision violation data. A plot of this data is show in Figures 6.28 and 6.29 for a trajectory height of 26 inches. This data corresponds to the same trajectory data shown for cost function 1. Because

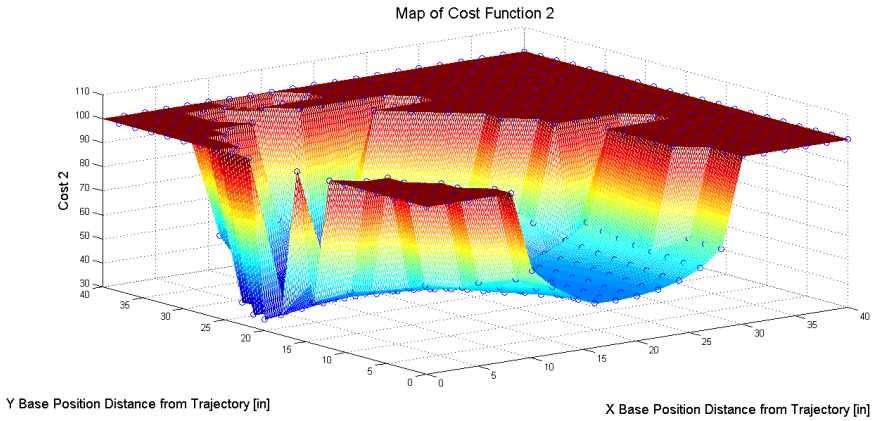

Figure 6.28. Plot of second cost function that penalizes shoulder movement during trajectory task.

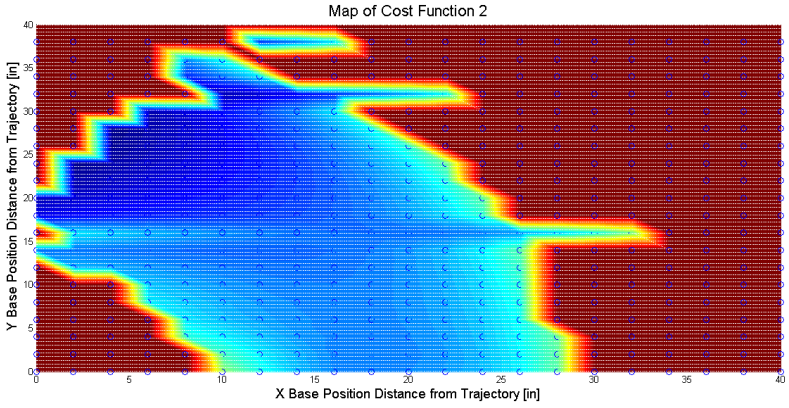

Figure 6.29. Aerial plot of second cost function that penalizes shoulder movement during trajectory task.

the trajectory is the same as before, all joint limits, collision, and joint velocity plots are the same. These plots are shown in Figures 6.9 through 6.18. Note that recalculation of the trajectories is not needed; the only thing that changes from one cost function to the next is how the joint motion data is combined into an overall plot of the optimal base position for the mobile robot.

6.2.8 Cost Function 3

The final cost function used for this analysis seeks to minimize power consumption of the robotic manipulator during a refueling task. This analysis looks at minimizing the maximum kinetic energy of the robotic arm as it performs a refueling task.

The kinetic energy of the manipulator can also be utilized to assess the dynamics of the system using the Lagrange equation. The Lagrange equation specifically looks at the difference between the kinetic and potential energy of a dynamic system to find the equations of motion for the robotic arm. Referenced in Jazar's text, "Theory of Applied Robotics", the kinetic energy is found using Equation 6.5 [18]. The kinetic energy equation uses the mass moment matrix and mass center angular velocity for each linkage on the arm. Figure 6.30 illustrates the mass center location for each corresponding link. The mass center was assumed to be located halfway down the length of each link. However for a more complete analysis, the mass center location would need to be determined experimentally for each linkage before the robot was assembled.

$$
\begin{aligned}
&K = \sum_{i=1}^{n} K_i = \tfrac{1}{2} \sum_{i=1}^{n} ({}^0v_i^T m_i {}^0v_i + \tfrac{1}{2} {}_0w_i^T {}^oI_{i0} w_i) \\
&{}_0w_i = \text{angular velocity in the global coordinate frame} \\
&{}^oI_i = \text{mass moment matrix in the global coordinate frame} \\
&{}^0v_i = \text{linkage mass center velocity in the global coordinate frame}
\end{aligned}
\tag{6.5}
$$

All mass and inertial properties were found by modeling each link in SolidWorks. The geometry of each linkage on the RE2 arm was measured and then created within SolidWorks. The material assigned to the arm was an aluminum alloy. The material used within the SolidWorks model sought to provide an estimate of the weight for each linkage. The total mass of the arm amounted to approximately 22.5 kg, which is around 50 lbs. Extra weight could be added to the modeled system to account for the weight of electrical wiring and motors on each joint based on physical measurements gathered prior to the arm assembly. The mass and inertial matrices for each link are referenced in Appendix B. For more detail on the derivation of each mass center velocity, please reference Appendix B section B.2.

Using Equation 6.5, the kinetic energy of the arm was calculated for each trajectory point. From the entire trajectory path, the maximum kinetic energy was selected for each (X, Y) base location. This maximum value was then assigned as the kinetic energy cost value for each base location. A plot of the maximum

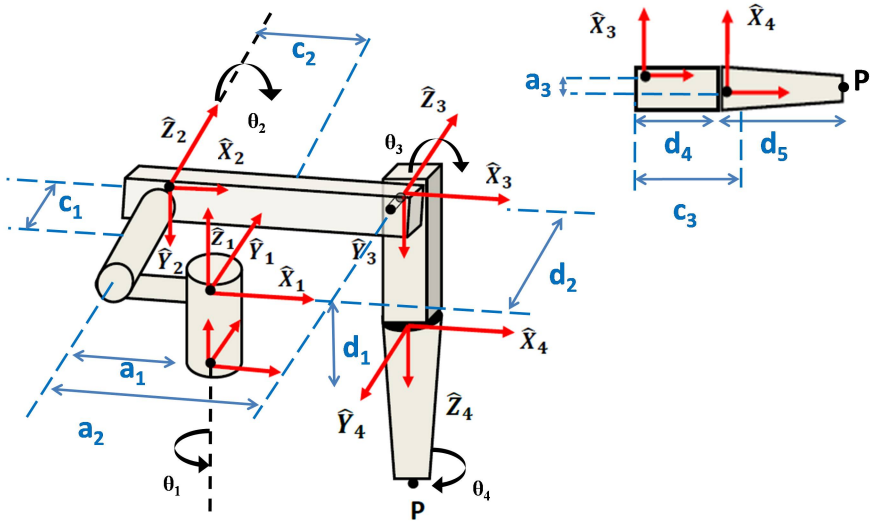

Figure 6.30. Diagram of RE2 robotic manipulator mass center geometry.

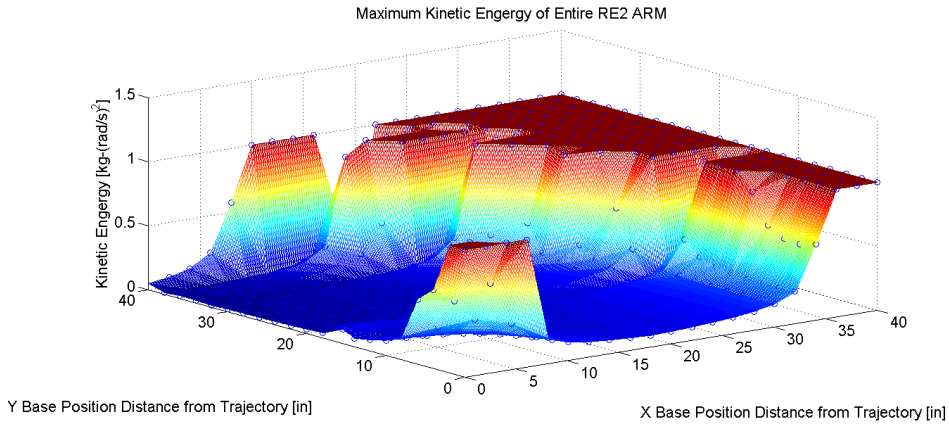

Figure 6.31. Plot of kinetic energy cost function at a reference trajectory height of 26 inches.

kinetic energy is shown in Figure 6.31. This plot calculated the kinetic energy for the reference trajectory at a height of 26 inches. Joint singularities that occurred at different base locations were assigned a cost value of 1, seen as high cost values in the outer perimeter of the 3D cost map. Looking at the 3D cost function map for kinetic energy, it can be seen that the shape of the basin is very similar to

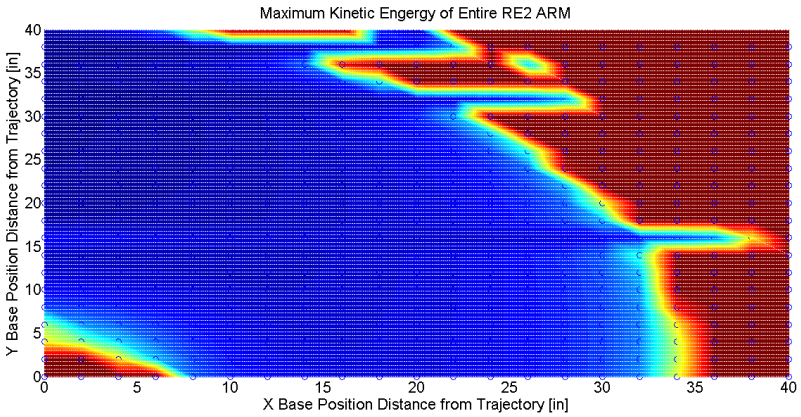

Figure 6.32. Aerial Plot of kinetic energy cost function shown in Figure 6.31.

the previous combined cost functions above. The similarities between these 3D cost maps occurs because the minimal angular velocities that occur for all joints illustrated in Figure 6.19 will also minimize the kinetic energy of the arm, as can be seen in equation 6.5.

Chapter 7

Optimal Base Location along a Trajectory Path

As the end effector of the manipulator travels a trajectory path, the robot can move forwards or backwards along the X or Y-axis to maintain the optimization metrics established by each cost function, represented by low cost values found on the 3D cost maps. Once these cost functions are established, optimal base locations can be chosen. In order to simplify the analysis, it is assumed that to perform a refueling task the robot will be aligned along either the X or Y-axis, allowing the robot to drive forwards or backwards while performing a manipulation task. This selection process is performed systematically by first establishing optimal base positions along the global X or Y axis for a corresponding fixed X, or Y value. In Figure 7.1 the mobile base moves along the Y axis as it follows the trajectory for a fixed X offset. Conversely, in Figure 7.2, the mobile base moves along the X-axis for a fixed Y offset.

7.1 Optimal Base Locations along Y-axis for a fixed X-offset

Starting with the configuration illustrated in Figure 7.1, the resulting optimal Y locations are plotted out for each fixed X-offset, ranging from 0 to 40 inches. These optimal locations are found using the overall combined cost function illustrated

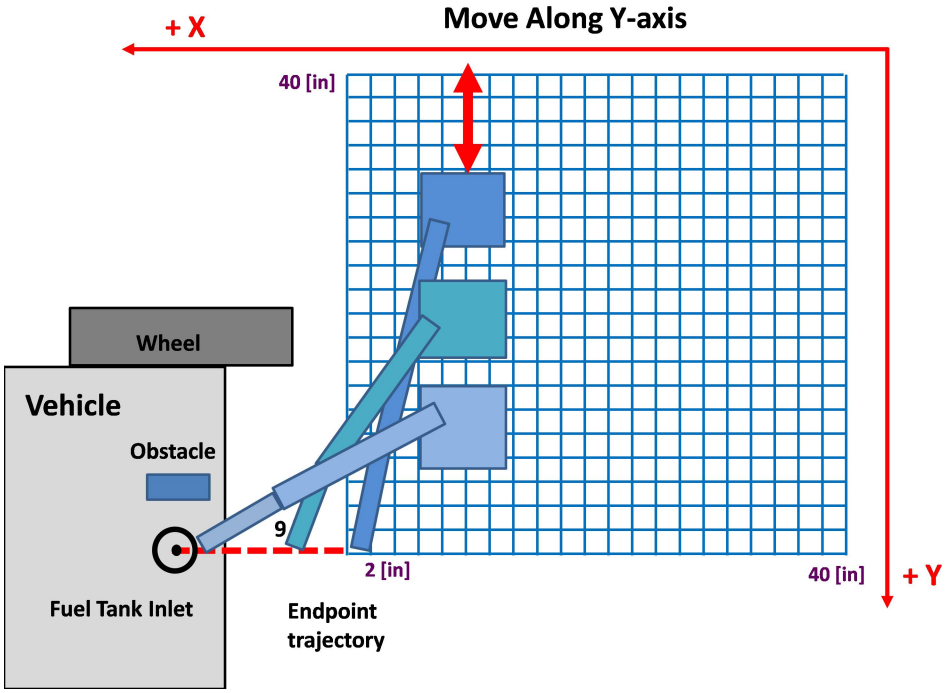

Figure 7.1. Illustration of mobile base movement along the Y axis with a fixed X offset while the end effector follows a commanded trajectory.

previously in Figure 6.19. Sample plots for X offsets of 0-4 inches, and 24 to 28 inches are shown in Figures 7.3 and 7.4. Looking at figure 7.3, you can see in the top plot that for an X-axis offset of zero inches, the optimal base location along the Y-axis is 20 inches throughout the entire trajectory.

Looking at an X value of 24 inches in figure 7.4, the optimal Y location consists of 20 inches for the beginning of the trajectory, and towards the end, the robot will move closer to the end effector path at a location of 16 inches. Depending on the location of the mobile base, the robot may move forwards and backwards along the Y axis several times to maintain optimization metrics as it performs a refueling task. From here, the robot can look at the optimal Y locations for each X offset away from the fuel tank inlet, and select the best X location.

The first selection process in choosing an X-offset consists of finding optimal base locations that minimize base movement. As seen in Figure 7.4, there are several X locations that have a single optimal base location throughout the entire trajectory. By minimizing base movement, external errors caused by wheel slipping

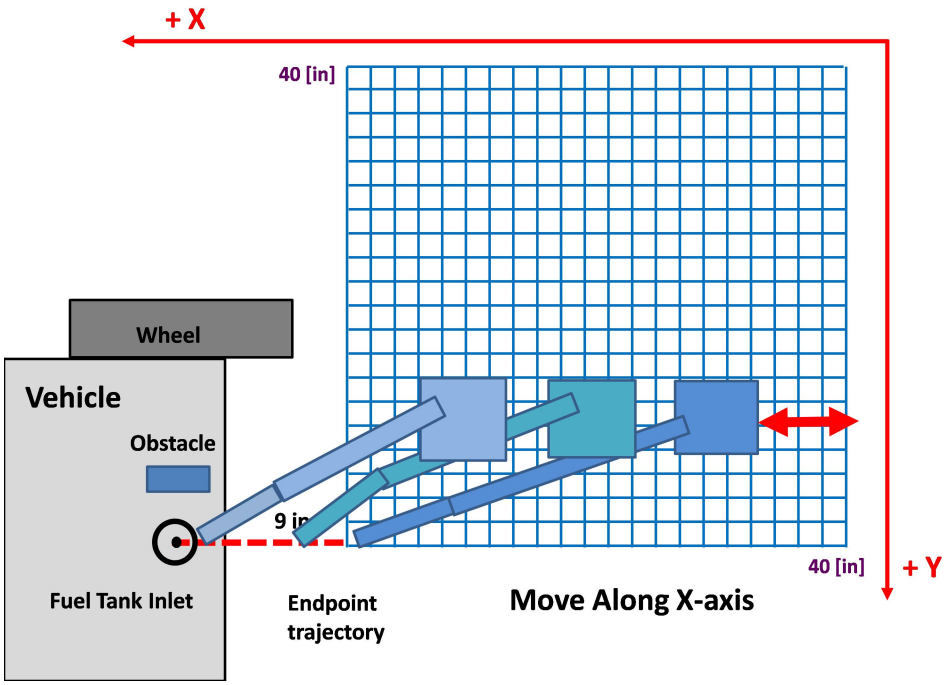

Figure 7.2. Illustration of mobile base movement along the X axis with a fixed Y offset while the end effector follows a commanded trajectory.

and shifting on uneven ground can be minimized by picking base locations that require minimal base movement for the refueling task. The selection of X positions that minimize base movement was done by summing the difference in Y location from one trajectory point to the next. Solutions that did not change base location would have a have a sum of zero. The more the base moves, the higher the total sum along the trajectory. Equation 7.1 illustrates how the sum of base movement is calculated.

$$Base\_Sum = \sum_{i=1}^{250} (abs[y_{optimal}(i) - y_{optimal}(i-1)]) \tag{7.1}$$

Referencing equation 7.1, a total sum of the difference in base location from one trajectory point to the next is calculated for points 0 to 250. This sum illustrates how much the base moves throughout the trajectory at a certain X-offset value.

The second step for choosing an appropriate X-offset location is to look at the list of locations that have minimal base movement, and choose an X location that

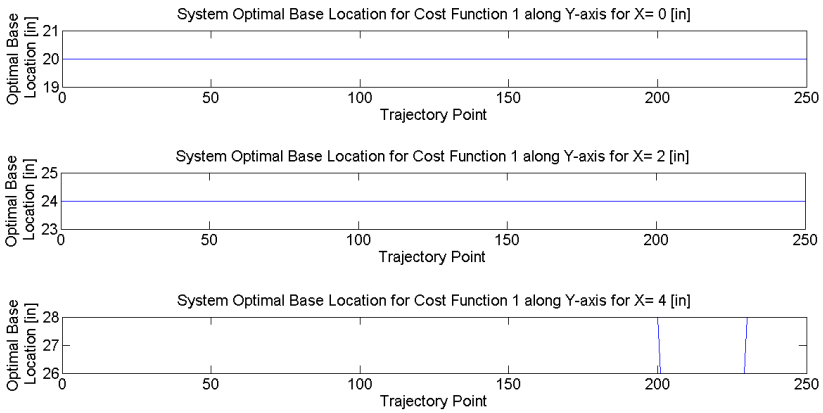

Figure 7.3. Optimal Y locations for X locations of 0-4 inches for 90 degree trajectory task at a height of 26 inches.

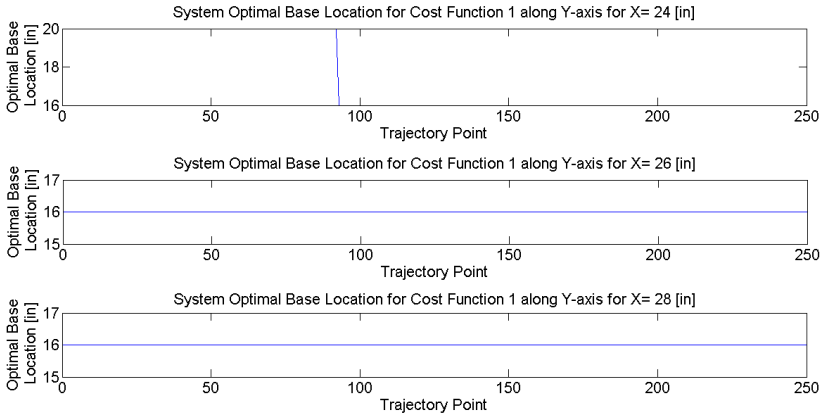

Figure 7.4. Optimal Y locations for X locations of 24-28 inches for 90 degree trajectory task at a height of 26 inches.

lies in the center of the low cost basin found in the combined cost plot in Figure 8.5. By choosing something that lies in the center of this basin, the arm has the ability to minimize its cost metrics while maintaining a safe distance away from singularities. Then if the robot does slip, or needs to move, all surrounding areas are safe for the mobile base to move to for adjustment during the refueling task. An aerial view of the combined cost function 1 discussed in section 6.2.5, is shown in Figure 7.5 with the optimal basin circled. Looking specifically at the X axis, there is a Y offset value where X has the largest range. This occurs at a Y offset value of 16 inches. Because the robot can move along the Y-axis but not the X, it is important to choose an X-offset that lies in the lower cost basin away from

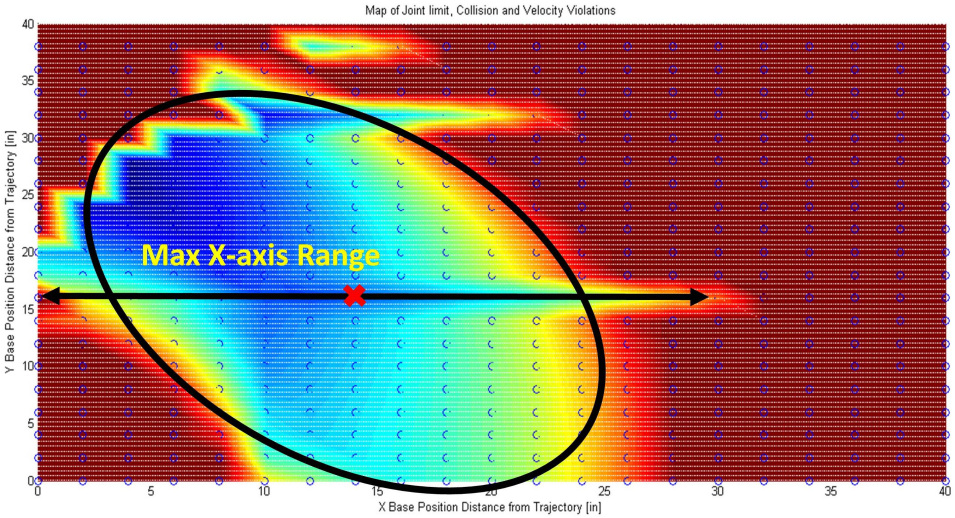

Figure 7.5. Illustration of optimal base locations for cost function 1 with a 90 degree trajectory setup at a height of 26 inches. Circle shows optimal base locations, and the line shows the maximum range of X locations. The center of this range serves as a metric for finding an optimal X location for the mobile base.

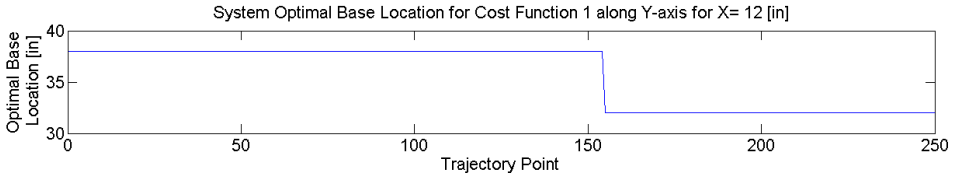

Figure 7.6. X location chosen for a refueling task at a height of 26 inches. The X location is at 12 inches, with a Y location of 38 inches along the Y axis for trajectory points 1-154, and a Y location of 32 inches for trajectory points 155-250.

singularities. An X-offset value near the center of the maximum X axis range was chosen for carrying out a refueling task. This center point is marked in Figure 7.5.

For this solution, X-offsets of 0, 2, and 26 inches had a single fixed location that remained optimal throughout the entire trajectory that had no base movement at all. From all base locations that are fixed, a base location of X=26 inches is the closest to the midpoint of the line marked in Figure 7.5 at 14 inches on the Y axis. Because this is still far from the central part of the basin, a criteria can be set to allow a total sum of 6 inches of wheel movement throughout the refueling task. With this criteria, a base location at X=12 inches is much closer to the center of the optimal basin. The Y-locations along the reference trajectory for an X-offset at 12 inches is shown in Figure 7.6.

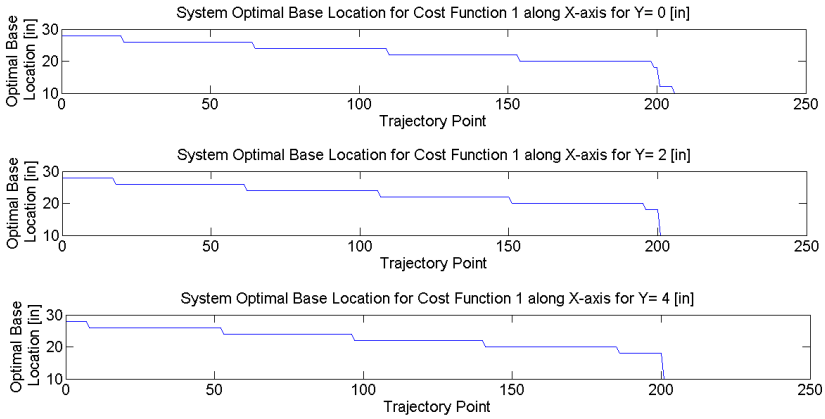

Figure 7.7. Optimal X locations for Y-offset values of 0-4 inches for a 90 degree reference trajectory at a height of 26 inches.

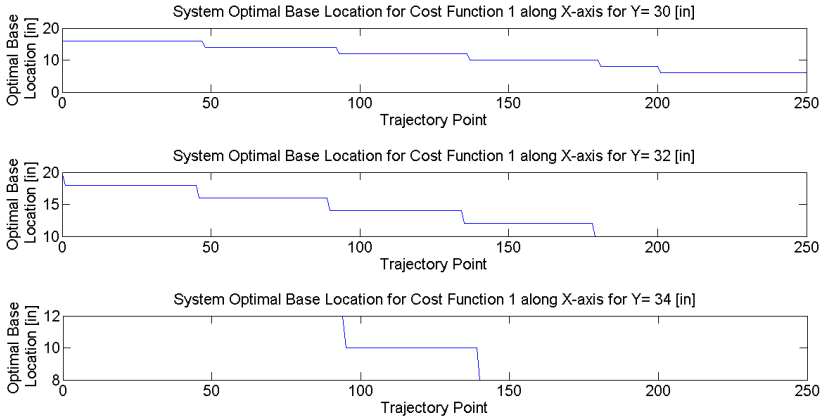

Figure 7.8. Optimal X locations for Y-offset values of 30-34 inches for a 90 degree reference trajectory at a height of 26 inches.

7.2 Optimal Base Locations along X-axis for a fixed Y-offset

The same process covered for base locations along the Y-axis was conducted for base locations on the X-axis. This configuration is illustrated in Figure 7.1. Figures 7.7 and 7.8 show optimal X base position values for Y-offsets of 0-4 inches, and 30-34 inches

Using the same methodology as above, the first step is to calculate a sum of the total base movement along the entire trajectory path for each Y-offset value from 0 to 40 inches. This means that each time the base changes position from

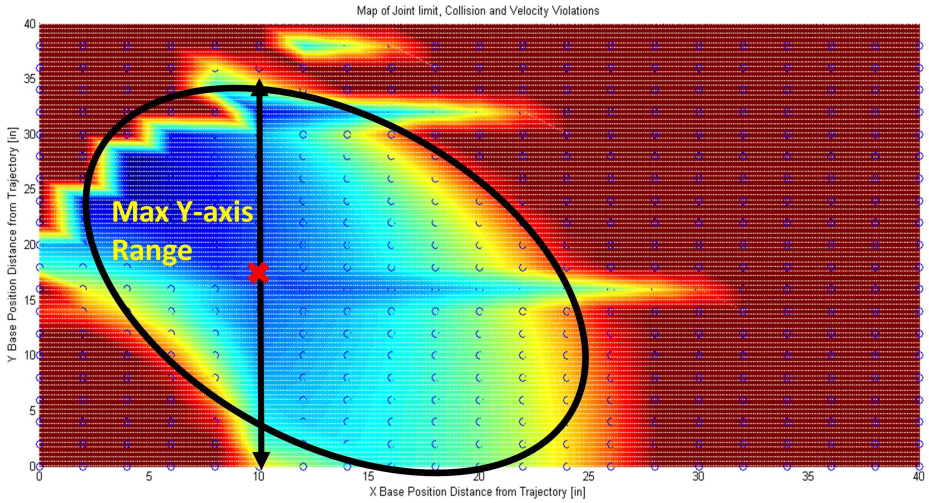

Figure 7.9. Illustration of optimal base locations for cost function 1 with a 90 degree trajectory setup at a height of 26 inches. Circle shows optimal base locations, and the line shows the maximum range of Y locations. The center of this range serves as a metric for finding an optimal X location for the mobile base.

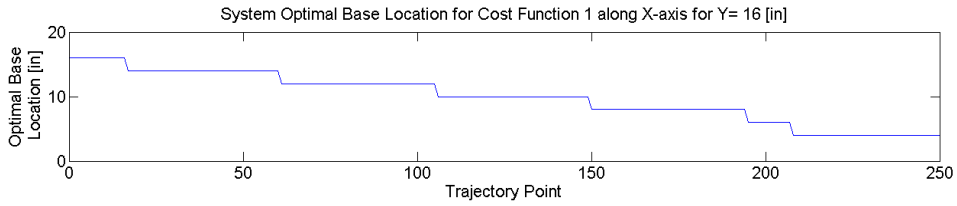

Figure 7.10. Optimal Y-offset of 16 inches was selected for refueling task. The corresponding X-axis base locations are plotted for the trajectory path.

one X location to the next, the difference between these two values is calculated and added to the total sum. This is described in equation 7.1. The more the base moves, the higher the total sum along the trajectory. Because movement along the Y-axis is much more frequent, a threshold of 12 inches is set. All values with a sum of less than or equal to 12 inches of base movement are considered as a desirable solution. These minimal base locations occur at Y offsets of 10, 12, 16, and 30-40 inches.

The second step is to choose from these Y-offset values a solution that lies closest to the center of the optimal basin. Looking at the maximum range along the Y-axis in Figure 7.9, the maximum range occurs at an X value of 10 inches. The center of this maximum range occurs at a Y-offset value of 19 inches. Because

the mobile base has the ability to move along the X axis, a Y offset is chosen so the center of the base is away from singularities along the Y-axis. This is done as a safety measure, because the robot can only move along the X-axis to adjust its optimal cost metrics, and not the Y-axis. So it is desirable to choose a safe Y-offset. The method for choosing from a list of Y-offsets is to calculate the distance between the Y-offset values, and the Y-offset that lies in the center of the minimal basin. This calculation is shown in equation 7.2.

Once the distance to this optimal Y-offset value is recorded for all selected Y-offsets, the value closest to the center is chosen. This is the value with the smallest sum using equation 7.2. For this example using Figure 7.9 as reference, a y-offset of 16 inches was the closest value to the center coordinate. A plot of the Y-offset of 16 inches and the resulting optimal X-locations throughout the trajectory is shown in Figure 7.10.

$$\begin{aligned} &\text{Distance from Center} = d_{i-center} = y_{\text{offset}}(i) - y_{\text{Basin Center}} \\ &\text{for i} = 1 \text{ to n}, \text{where n is the number of Y-offset solutions} \end{aligned} \tag{7.2}$$

7.3 Joint Velocities for Optimal Base Locations

Corresponding joint angles and joint velocities of the selected X, and Y offset positions are plotted in Figures 7.11 through 7.14. For mobile base movement along the X-axis, as illustrated in Figure 7.1, an X-offset of 12 inches was chosen, where the optimal base locations are plotted in Figure 7.6. The corresponding joint angles and joint velocities of the robotic manipulator are plotted in Figures 7.11 and 7.12. Comparing the base positions for an offset of 12 inches in Figure 7.6 to the joint angles in Figure 7.11, you can see that when the base moves at trajectory point 155, there is a jump in the joint angles as well. This jump can easily be smoothed out by gradually moving the base forward along the trajectory, smoothing out the base position plot in Figure 7.6.

Next, joint angles and velocities are plotted for the base movement along the Y-axis configuration shown in Figure 7.2. For this configuration, the Y-offset value of 16 inches was chosen, plotted in Figure 7.10. The corresponding joint angles and joint velocities are shown in Figures 7.13 and 7.14.

Looking at the results for a robotic manipulator traveling along the X-axis to

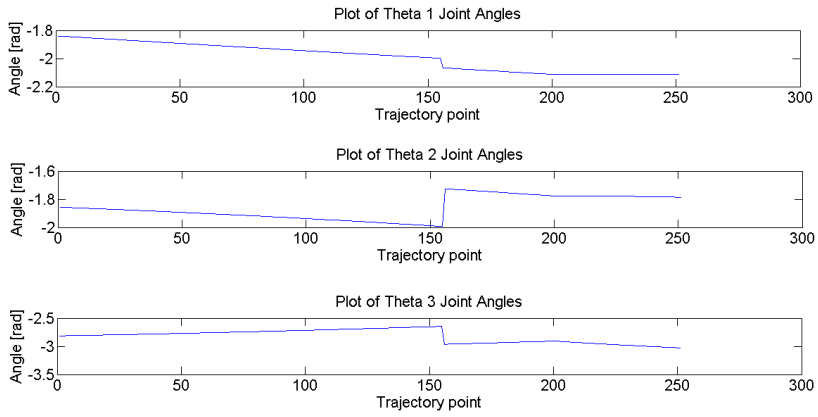

Figure 7.11. Joint angles for refueling task at base locations shown in figure 7.6. Plots of joint angles are for yaw-θ_1, shoulder-θ_2 and elbow-θ_3 joints; the discontinuous joint angles at trajectory point 155 is due to a change in base position.

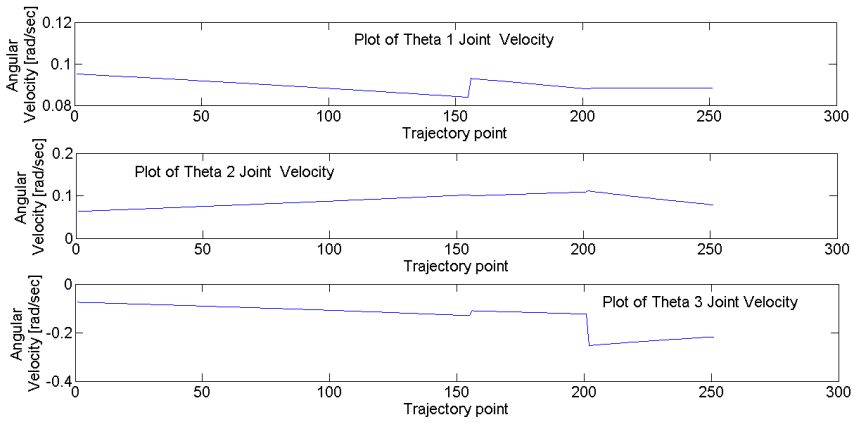

Figure 7.12. Plot of yaw-θ_1, shoulder-θ_2 and elbow-θ_3 joint velocities along trajectory path corresponding to joint angles and base positions shown in Figures 7.6 and 7.11.

approach the fuel tank, it seen that it is necessary for the robot to move forward several times throughout the trajectory task. If the robot approaches a fuel tank from this configuration, the robot will utilize the mobile base movement as well as the movement of the manipulator arm. Because of this, it is important to coordinate the mobile base and arm movement together when planning a refueling task.

The basin of available base locations can also yield insight into mobile base design, specifically for defining necessary tolerances on wheel location, and acceptable error in global position estimation from external sensors such as stereo vision,

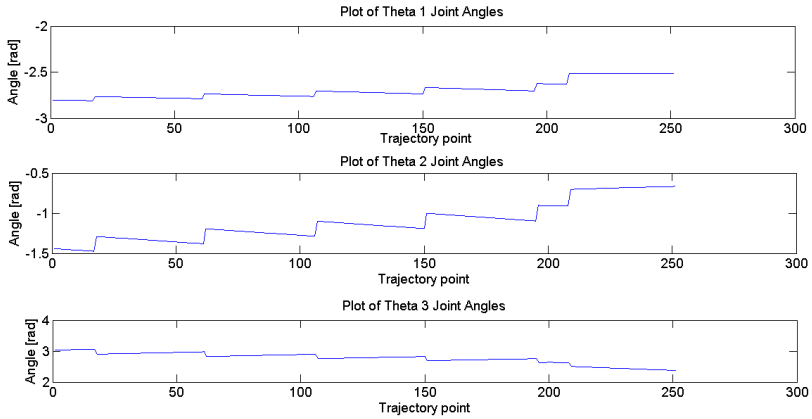

Figure 7.13. Joint angles for refueling task at base locations shown in figure 7.10. Plots of joint angles are for yaw-θ_1, shoulder-θ_2 and elbow-θ_3 joints.

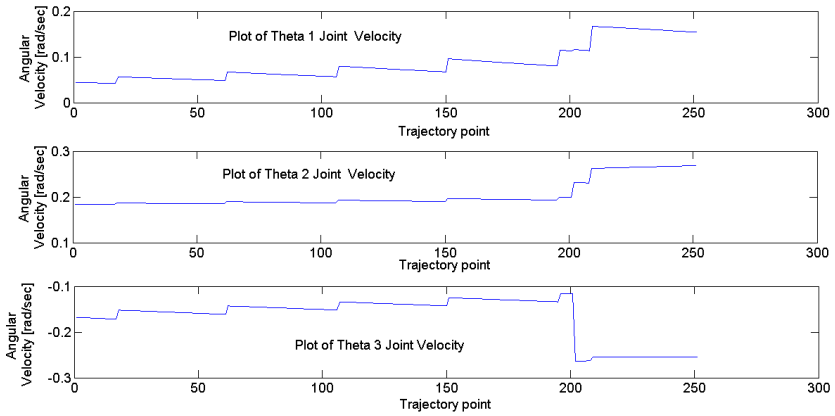

Figure 7.14. Plot of yaw-θ_1, shoulder-θ_2 and elbow-θ_3 joint velocities along trajectory path corresponding to joint angles and base positions shown in Figures 7.10 and 7.13 .

LIDAR or global GPS location. Based upon the plots shown in this section and the next, the robot will have an idea of where it can navigate to in order to optimize pre-determined constraints before the refueling task occurrs. The robot will also have the ability to predict where best to position itself based upon specific linkage dimensions, and mobile base height relative to the desired trajectory. With these metrics in mind, the robot will choose a position where it can carry out a successful refueling task even if the mobile base was to shift while performing a task.

Chapter 8

Optimal Trajectory Height

In addition to mobile base location, a second constraint must also be considered when looking at mobile base design for the RE2 arm. Variations in the height of the end effector trajectory in relation to the origin of the arm can greatly affect the available manipulator operating space. This is a primary driver to be considered when designing the mobile base.

This section focuses on the relationship between the height of the trajectory path and the available solution space of the mobile base position. For each cost function there is an optimal height that maximizes the available space for the manipulator position. This should be a primary design goal for the mobile base of any refueling robot. A wider solution space for a refueling task allows for more choices in robot navigation to avoid uneven and rough terrain near the fuel tank, larger tolerances in positioning control, and more flexibility to position the robot away from any obstacles that may affect entry into the fuel inlet.

Aerial plots of the combinded 3D cost function 1 from section 6.2.5 are shown in Figures 8.1 to 8.9 of trajectory heights ranging from 42 inches to 10 inches above the origin of the robotic arm. Negative heights were not considered for this analysis because heights closer, or below the origin of the robotic arm would imply that the mobile base would need to be as high (or higher) than the refueling inlet. This situation was not considered strictly because the fuel inlet on several military vehicles is several feet off the ground, and a mobile base of that height would lead to stability problems in navigation. It could also introduce more collision possibilities if more objects extrude from the vehicle at larger heights.

Figure 8.1. Aerial plot of cost function 1 for a trajectory height of 42 inches.

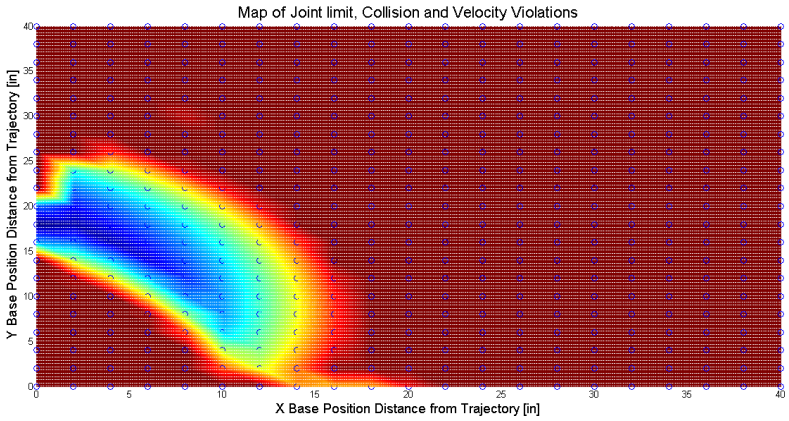

Figure 8.2. Aerial plot of cost function 1 for a trajectory height of 38 inches.

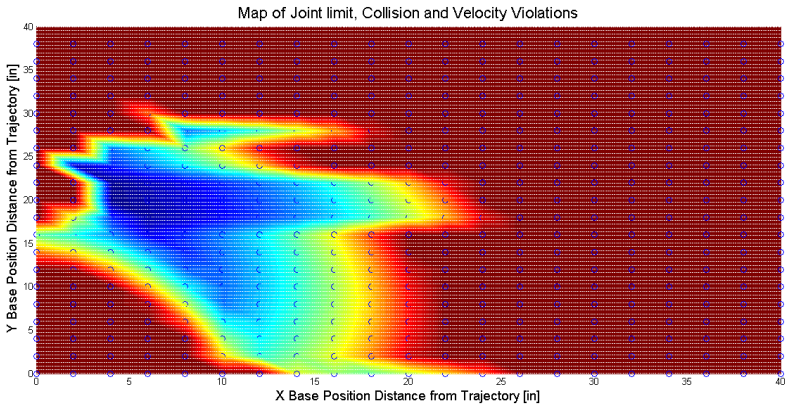

Figure 8.3. Aerial plot of cost function 1 for a trajectory height of 34 inches.

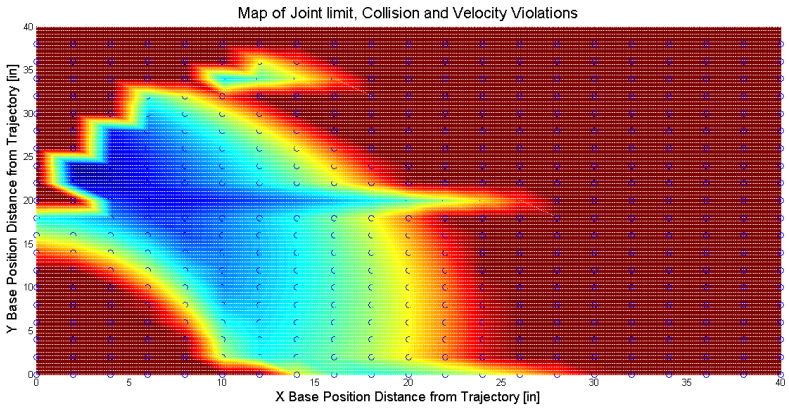

Figure 8.4. Aerial plot of cost function 1 for a trajectory height of 30 inches.

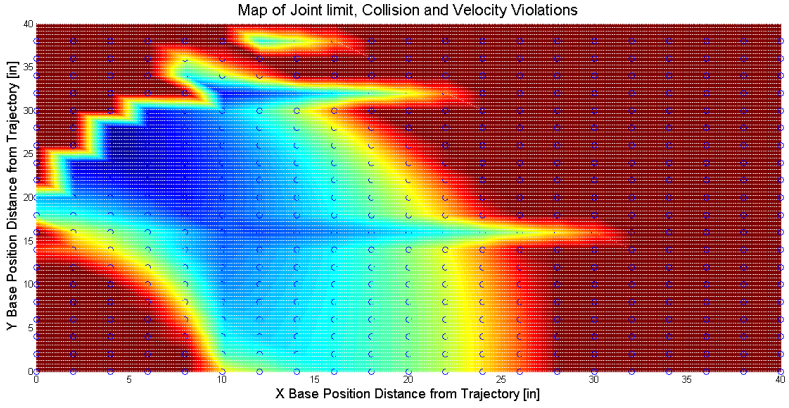

Figure 8.5. Aerial plot of cost function 1 for a trajectory height of 26 inches.

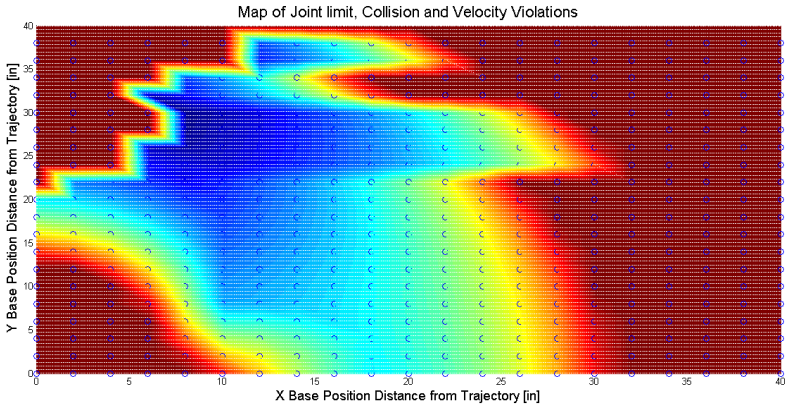

Figure 8.6. Aerial plot of cost function 1 for a trajectory height of 22 inches.

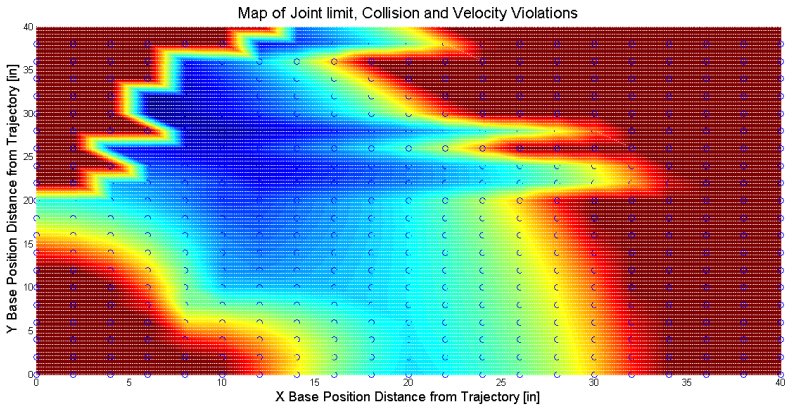

Figure 8.7. Aerial plot of cost function 1 for a trajectory height of 18 inches.

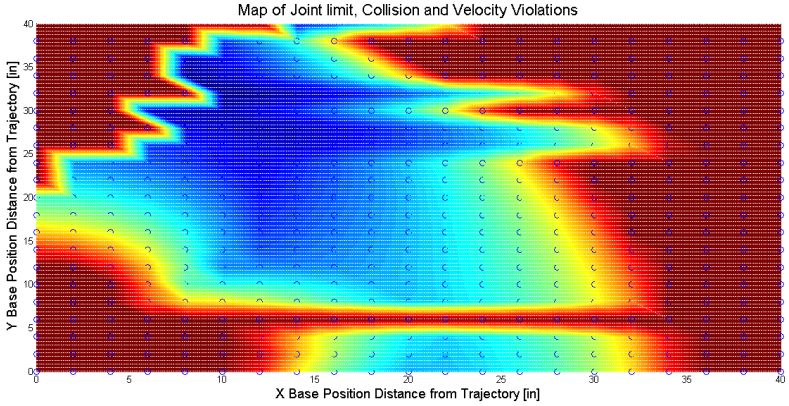

Figure 8.8. Aerial plot of cost function 1 for a trajectory height of 14 inches.

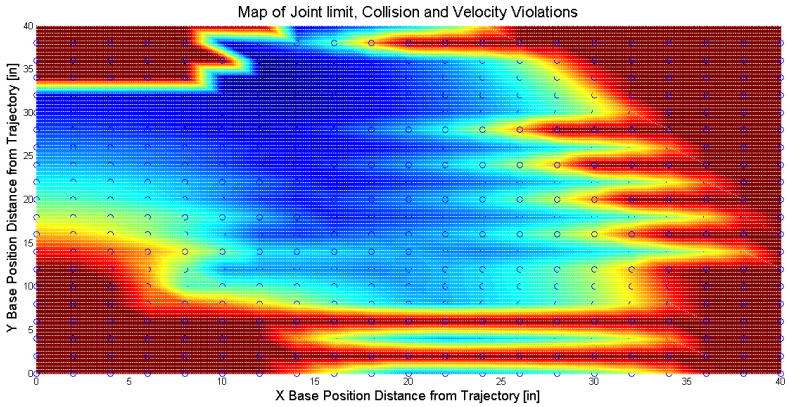

Figure 8.9. Aerial plot of cost function 1 for a trajectory height of 10 inches.

Looking at Figures 8.1 through 8.9, it can be seen that as the height between the origin of the manipulator and the end effector decreases, the amount of available space the robot can navigate to increases. The red violation lines that occur at lower Y values for Figures 8.9 and 8.8 are joint limit violations that occur when the arm converges to an elbow down configuration, or the shoulder moves too far back violating the limits for hitting the tooling platform. At some point, as the trajectory height decreases, it will become more difficult for the arm to perform a refueling task for risk of kinematic lock up that occurs for trajectory points close to the manipulator origin. It was found that there is an optimal height for each cost function that maximizes the space available for the mobile manipulator to operate in. For the first cost function, this occurs at a height of 18 inches. At lower Y positions, singularities and joint limit violations occurred for heights of 10 and 14 inches, some of which was because the Newton-Raphson solver was unable to converge to an elbow up configuration. Further work could be done to improve the robustness of this solver; however, with the current setup, the height of 18 inches appears to be optimal.

Figure 8.10 shows the entire available space for the robotic manipulator, and one can observe that there is symmetry in the cost function about the X-axis. This symmetry was expected; therefore the grid of base locations shown in Figure 6.6 was calculated only on one side of the trajectory path, knowing that everything on the other side of the trajectory would simply be a mirror image of the first solution space. One exception to this assumption would be extruding obstacles that occurred on one side of the fuel inlet, and not the other. For this case it is assumed that the obstacle is the same on both sides of the gas tank. If the obstacle were not identical on each side, another cost function would need to be created from the opposite side accounting for other obstacles.

Figure 8.11 illustrates the scale of the available space for the manipulator for the 90 degree refueling task. For a trajectory height of 18 inches, the available space along the Y axis is around 5 feet long, and 2 feet wide along the X axis, making a ring of optimal base positions around the desired trajectory. This process was repeated for each cost function to establish an optimal trajectory height.

By understanding how the trajectory height relative to the origin of the robot manipulator affects the size of the low-cost basin, a mobile base height can be

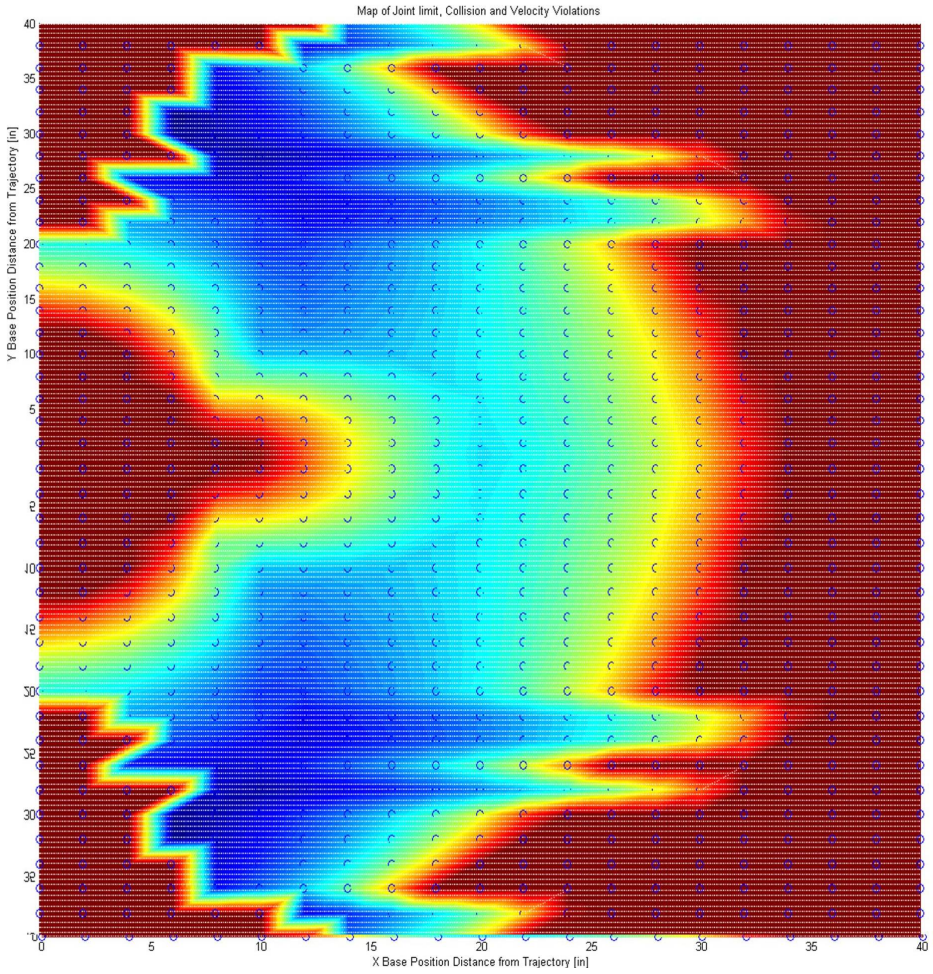

Figure 8.10. Figure of optimal trajectory height of 18 inches, reflected across the X axis to show the full available space.

selected that optimizes the cost basin shown in Figure 8.10. Because there may be different fuel inlet heights on different vehicles, a best height may be chosen that maximizes the available space across multiple vehicles. Looking ahead at Table 10.1, the fuel inlet height on two sample trucks was around 35 and 51 inches off the ground. As seen in figure 8.7, an optimal trajectory height of 18 inches relative to the origin of the manipulator would require a mobile base height of 33 inches for a gas tank 51 inches off the ground. Looking at the lower fuel tank height of 31 inches, the arm would only need to be 13 inches off the ground to maintain a distance of 18 inches between the origin of the arm and the fuel tank inlet. This means that the mobile base height can be anywhere between 13 to 33 inches tall.

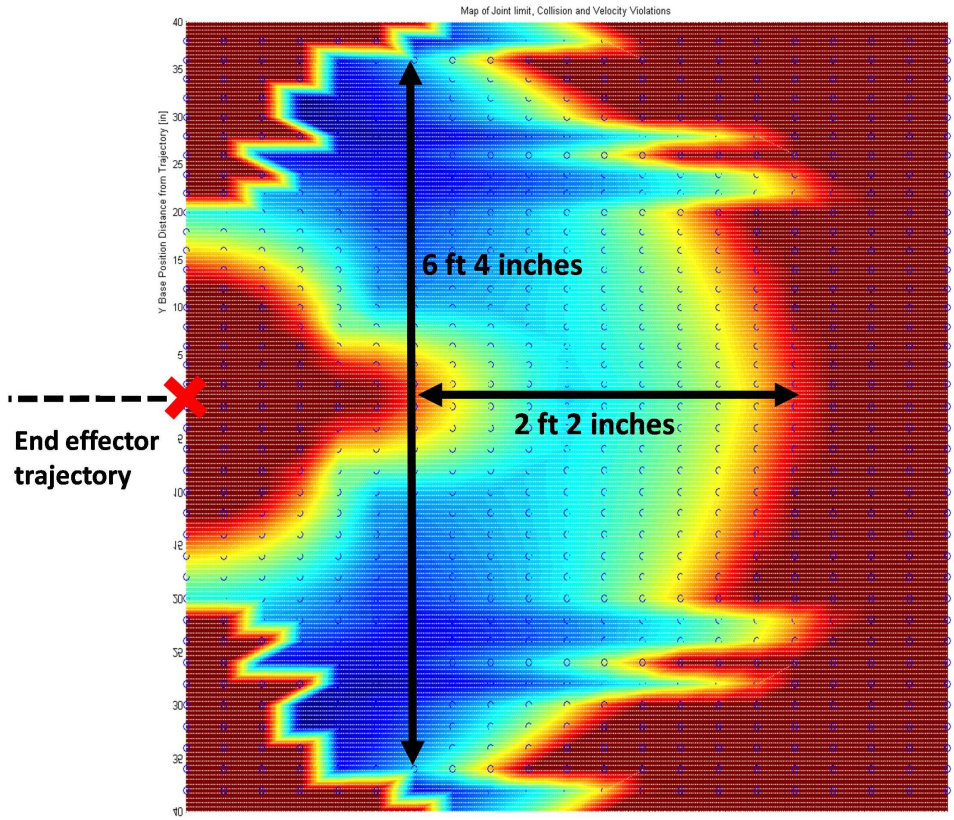

Figure 8.11. Aerial view of full available space for a trajectory height of 18 inches.

To accommodate both these heights, a base height could be somewhere in the middle of this range.

Chapter 9

Modeling and Simulation of RE2 Robotic Arm and Base

To test the motion and control of the arm in a virtual environment, a 1:1 scale CAD model of the RE2 arm was created and imported into a GAZEBO simulation package. GAZEBO is a robot simulator that allows simulation of physical motion and sensing algorithms in a three-dimensional world [20]. GAZEBO models were created to safely test the movement of the arm with a new control system before physically testing the RE2 arm, to confirm that the motion commands developed in previous sections were suitable for operation of a real-world arm. Sample gas tanks and vehicle models can be imported into the virtual world to provide full testing of the arm navigation and manipulation to place a refueling tube into different sample tanks. Joint positions, velocities, and accelerations are published from GAZEBO, as well as collision information between defined objects, and can be compared to the commanded joint motion as well as to the MATLAB simulation results developed earlier. The ray-casting script discussed in Chapter 6 can be used offline to filter out base positions that lead to collisions, allowing only safe solutions to be simulated within GAZEBO. GAZEBO also has the ability to detect any collisions that occur between an imported robot and any obstacles around it within the GAZEBO world. However this collision detection is only internal within GAZEBO, and a ray-casting script in combination with external sensors would need to be used with a robot performing a physical refueling task.

There are many advantages in simulating the robotic system within GAZEBO.

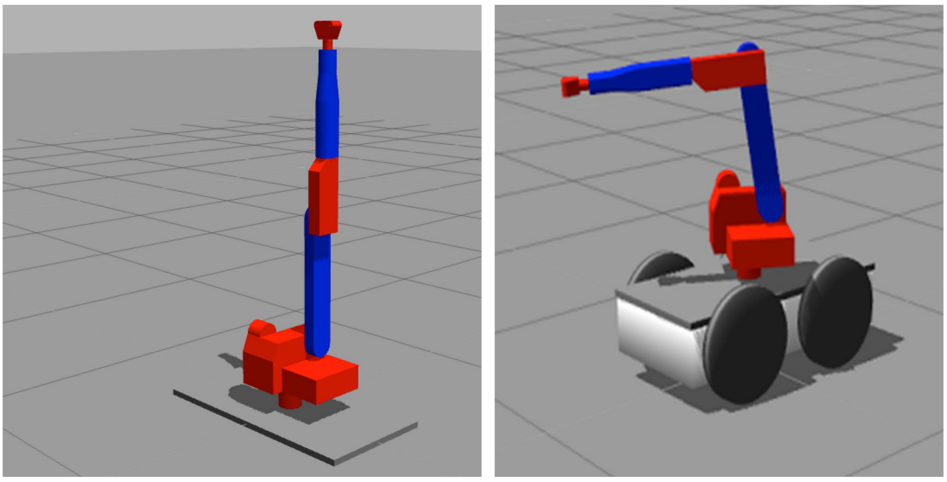

Figure 9.1. GAZEBO models of RE2 arm with fixed and mobile base.

For example, the control system designs can be tested first within the virtual environment to ensure there are no bugs within the software that could damage the RE2 arm. The user also has the ability to design a mobile base and add degrees of freedom to the arm to test how these extra joints can benefit a given task. In addition to adding degrees of freedom to an existing robot, other mobile robot models can be imported into a GAZEBO world. These models can be tested to assess the robot's ability to perform refueling tasks. These tests can be used as a method of comparison between different models.

Two different GAZEBO models were created to carry out refueling test, the first with the RE2 arm fixed to the ground, and the second with the same arm mounted on a mobile base. Figure 9.1 shows both configurations, and Figure 9.2 shows a screen shot of the robotic arm performing a refueling motion on a sample vehicle. This model was based on the Stryker fuel tank configuration shown in Figure 9.3.

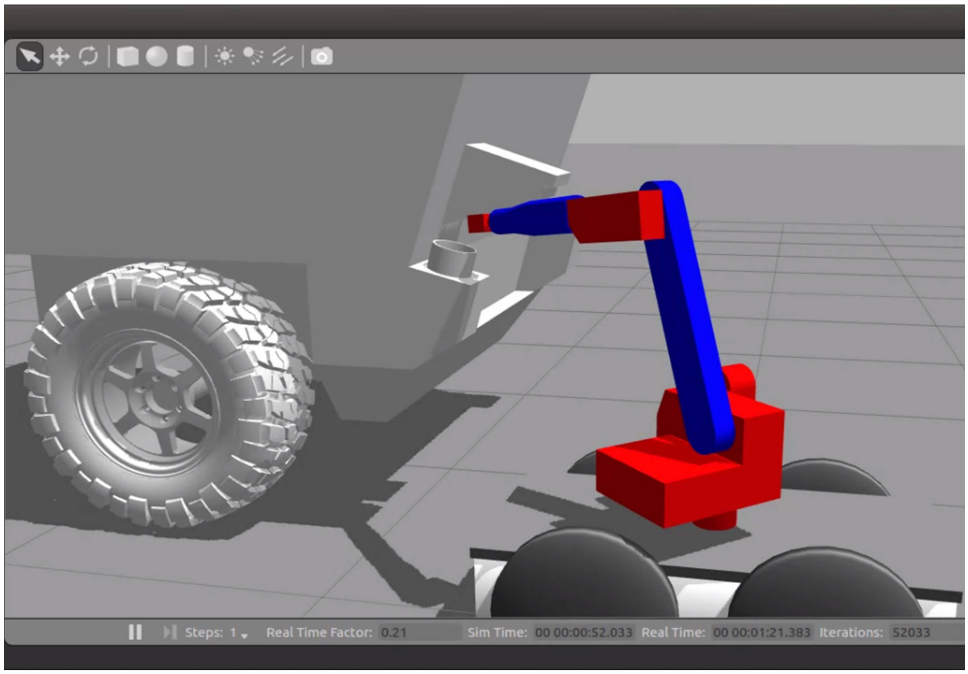

Figure 9.2. Screenshot of refueling test on sample vehicle captured.

Figure 9.3. Stryker fuel tank configuration used in ROS simulation.

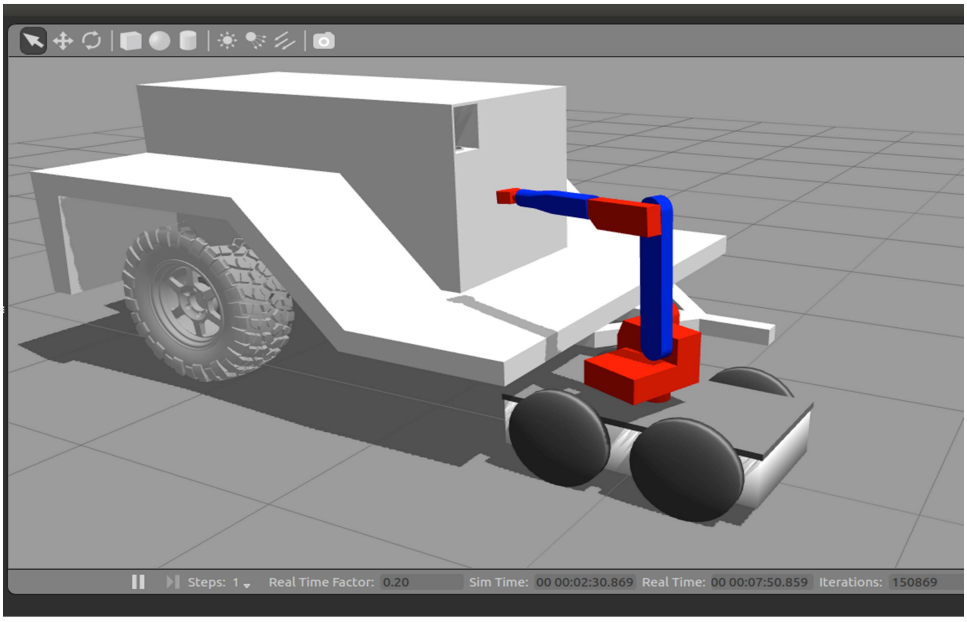

Figure 9.4. Screen shot of RE2 arm simulation in ROS with generator mounted on a trailer. The generator fuel port is located on the left corner.

Figure 9.5. Picture of generator mounted on trailer used as reference in ROS simulation.

9.1 ROS Communication

The GAZEBO system communicates with ROS using a package called ROS control [21]. This package provides a generic PID controller that moves the simulated actuators at each joint on the robot. The actuators within the GAZEBO world are controlled by joint effort controllers available within the ROS control package. An expression for joint position will be sent to the ROS controllers, and a command will be sent to the joints within the GAZEBO simulation. The joint will be controlled by a position controller, wherein the PID gains of the system can be adjusted within the ROS control configuration file. Details regarding the file layout of the ROS/Gazebo system are found in Appendix section A.1. An inverse kinematic node written in C++ was created to provide all the inverse kinematic computations for the appropriate joint angles to follow a desired trajectory path. Once these reference joint angles were acquired, the joint angles were published in a ROS topic message that was then utilizedby the ROS controllers. These controllers would then communicate with GAZEBO to simulate the movement of the joints of the RE2 arm CAD model. The communication scheme between ROS, GAZEBO and the Inverse Kinematic node is outlined in Figure 9.6.

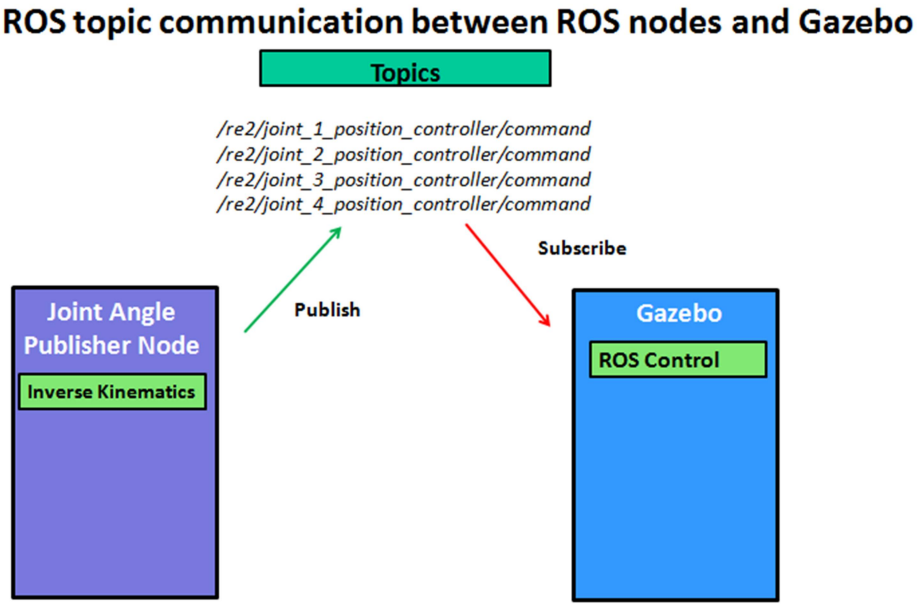

Figure 9.6. Illustration of communication between Inverse Kinematic node that publishes joint angles, and the GAZEBO simulation system.

Chapter 10

Testing and Validation

Previous sections described the steps taken to provide a kinematic understanding of the refueling task, and the development of both hardware and software infrastructure to successfully execute the refueling tasks. This section describes the simulated refueling tests that were conducted using the RE2 arm on several gas tank models with the fixed and mobile base configuration.

10.1 System Setup

A new operator control unit (OCU) for the RE2 automatic arm was developed as part of this study that provides closed loop control of the robotic arm joints using both angle and Cartesian control of the arm end effector in X, Y, Z space. The arm communicates via standardized JAUS messages with the OCU, which has been fully integrated with ROS (Robot Operating System) [3]. The ROS system allows a simple interface for message passing and playback of messages in communication with the OCU, the robotic arm, and the mobile base. ROS also allows easy access to sensor drivers, and links to command line tools such as RVIZ and GAZEBO used for modeling and simulation described in previous sections [21].

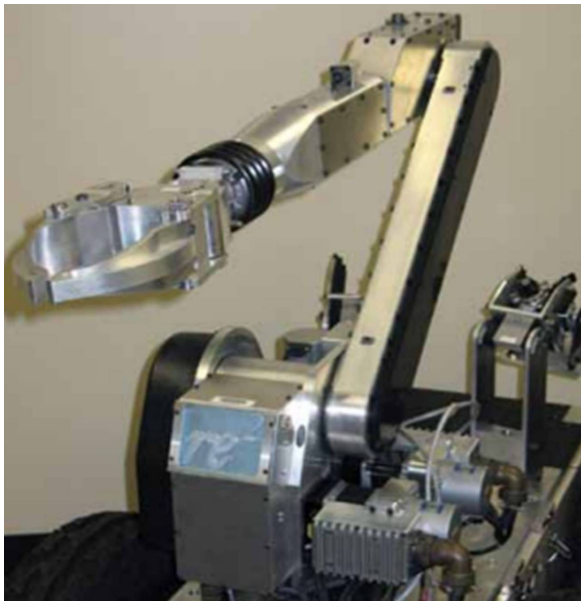

Figure 10.1. Picture of RE2 Automatic Arm used for testing. Photo credit Jesse Pentzer.

10.2 Control System Algorithm for a Fixed Base Robotic Arm

A general outline of the fixed base control system for the RE2 arm is shown in Figure 10.2. This controller consists of a goal orientation that is fed into an inverse kinematic calculator that solves for each joint angle using the Newton-Raphson iterative method. Once the appropriate angles have been computed, the joint angles are then fed into a PID (Proportional Integral Derivative) controller, with the derivative terms sent through a low-pass, second order Butterworth filter to reduce noise.

Based on performance results from testing the robotic arm along a desired refueling trajectory, a camera feedback loop may be incorporated into the control system to reduce error in the position of the end effector. This feedback loop is shown with the blue dotted line in Figure 10.2. The camera can be placed at a convenient location on the last link of the arm or on the end effector.

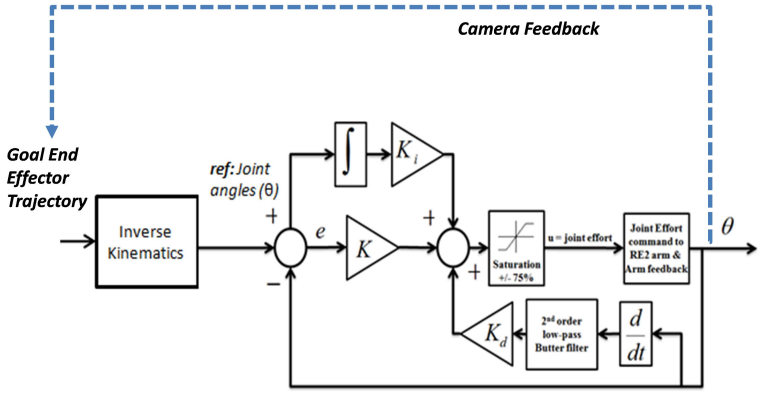

Figure 10.2. RE2 arm control system for fixed base.

10.3 Control System Algorithm for Robotic Arm with Mobile Base

Incorporating base movement within the refueling control loop allows the robot to adapt to uncertainties that occur in a range of different environments. For example, in uneven terrain a robotic manipulator may follow a refueling trajectory based on open-loop commands or on closed-loop feedback of joint positions. During operation however, errors in the pose estimate of the mobile base may cause the end effector to fall short of the goal position. Inside the trajectory control loop, the robot can choose to move the base or arm forward to account for base pose positioning errors that joint-position sensors could not detect. These pose errors could be caused by wheel slip in difficult terrain, sensor noise, or many other external factors. Using the optimal base solution found in Chapter 6, the robot may be required to move forward or backward to maintain the optimization metrics defined in a specific cost function.

The outline of the control system for the mobile robotic arm is shown in Figure 10.3. For this system, the goal position is fed into a decompose movement subsystem that mathematically breaks the coupled motion into an arm goal pose and base goal position. This de-coupling is based upon the mathematical rela-

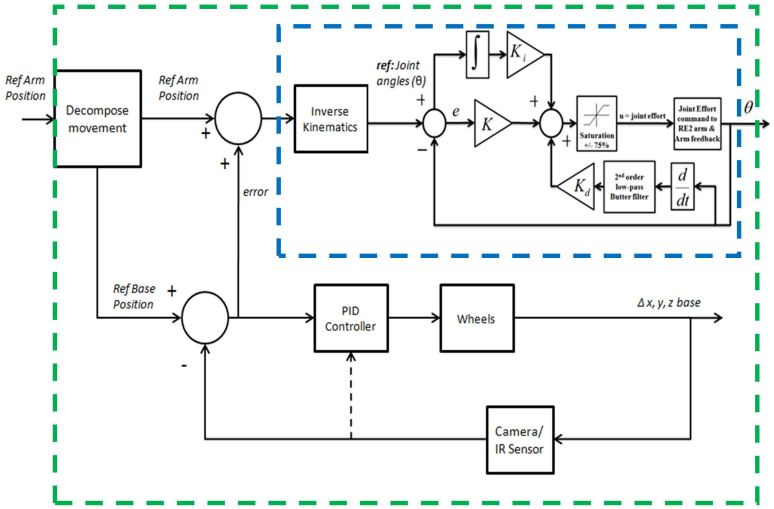

Figure 10.3. Overview of control system for mobile base and robotic arm.

tionship between the manipulator and base joints. The de-coupling of movement between the arm and the base was approached in two different ways. The first approach was to consider the mobile manipulator as a static entity that remained at a certain fixed location while refueling a vehicle gas tank. This method was used to analyze the base location along a grid of possible locations in Chapter 6. An optimal base location was then found for each trajectory point resulting in a series of optimal base locations for each end effector point along the path. This process has the ability to predict where the mobile base needs to be without fully analyzing another degree of freedom, thus mimicking mobile base movement. Therefore, if an incorrect base position is detected, the first approach performs a one-time movement to correct the base location to the re-calculated optimal base location.

The second method for characterizing the movement of the arm and the base is to move the base during motion of the arm, to compensate for base errors. This was modeled mathematically by adding a prismatic joint to represent base movement of of the robotic manipulator. This mobile base model was described in section 5.3, where the forward kinematics and Jacobian of the system was derived. The appeal of this second method was to use the inverse kinematics of this solution to solve for the mobile base velocity in addition to the desired base position, so that any

corrections could be performed smoothly during normal operation. This approach, however, results in a mathematically underdetermined system, meaning it has more variables than constraints on the system. In order to solve the resulting equations, a fourth constraint equation must be implemented. As a solution to this problem, optimal base positions were fed into the forward kinematics of the system to solve for the Jacobian. Referencing equation 5.4, the joint velocities can be solved for by multiplying the global X, Y, Z velocities by the inverse Jacobian. However because the Jacobian was not square, the pseudo-inverse was used to solve this equation. The pseudo inverse used to solve an underdetermined system is shown in equation 10.1.

$$\dot{\theta} = J^T(JJ^T)^{-1}\dot{q} \tag{10.1}$$

Equation 10.1 finds the least-norm solution of the pseudo-inverse [22]; however, when this was applied to find the base velocity of the robotic system, the resulting velocities were not optimal for the robotic arm. To test the pseudo inverse, already solved joint angles and joint velocities were selected for a base configuration where it was optimal for the robot to stay at a static location throughout the trajectory. Using the pseudo inverse, the joint and base velocities converged to an alternate solution that produced movement in the mobile base, and adjusted the joint velocities of the arm. Although this solution was correct, it was already determined that this solution was not the best solution.

Because the second method produced non-optimal velocity solutions, the first method was chosen to choose base location and joint velocities. Using the first method, the mobile base can be controlled by a position controller that receives a goal position for the mobile base found from the cost function plots. Using the current mobile base location, the resulting end effector goal location and mobile base origin were fed into the inverse kinematic node. The inverse kinematic calculator is then used to calculate output reference joint angles for the arm that were sent to the same PID controller as shown in the fixed base control algorithm.

The desired translational command to the mobile base can be regulated by a PID controller to move the four separate wheels of the base. Feedback to the mobile base controller can be provided by encoders and aerial cameras. This feedback can also be sent to the pose estimation for the robotic arm, so the arm can adjust

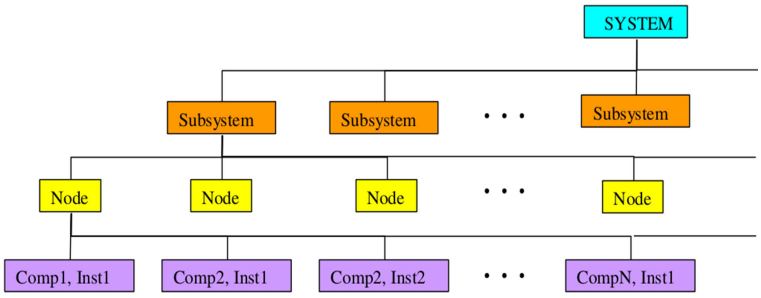

Figure 10.4. JAUS topology of system, subsystem, node and component hierarchy within the JAUS message structure [3].

its orientation based upon the movement of the base and the location of the gas tank. The entire control algorithm of the mobile robotic arm is outlined in green in Figure 10.3. The fixed base control system is outlined in blue.

10.4 ROS-JAUS Communication

The RE2 Automatic Arm uses standardized JAUS messages from OpenJaus Reference Architecture version 3.3 to send and receive data to and from robot arm [3]. The JAUS framework consists of a hierarchy of Systems, Subsystems, Nodes, Components and Instances. Figure 10.4 shows the topology of the JAUS architecture illustrated in the V3.3 JAUS Reference Architecture Specifications, Volume II Part I [3]. In this case, an example of a system is the manipulator arm itself. Subsystems of the arm consist of specific functions that may occur, such as sensing obstacles using sensor input, or a subsystem that processes data and navigates up to a specific location. Further functionality down this hierarchy consists of specific drivers that can operate sensors, actuators or other mechanisms attached to the robot.

The RE2 Automatic arm is controlled by an outside ROS Arm Controller Node

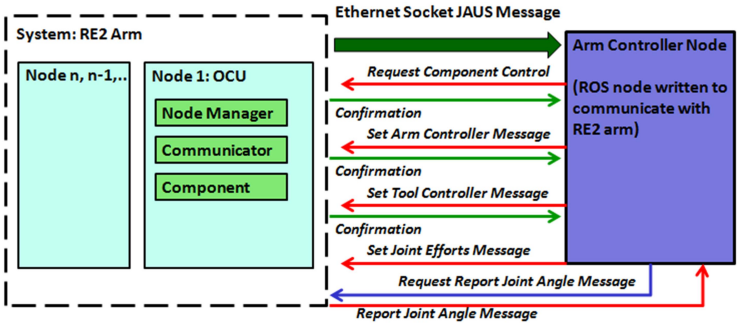

Figure 10.5. Message exchange between arm controller node, and OCU of RE2 arm for initial setup.

that communicates with the arm OCU using the framework referenced in the JAUS topology. The arm controller node sends a series of messages requesting component control, set joint efforts, and then reports commanded joint angle messages for full control of the robotic arm. These messages are sent to the OCU of the RE2 arm, via standardized JAUS messages [3], [23]. Figure 10.5 illustrates the message exchange between the Arm Controller Node and the RE2 arm OCU in order to send and receive joint messages and commands.

The controller node also controls the arm itself. Within the Controller Node, there are several different modes that have been created to control the arm in open loop. Joint effort control allows the joints to be controlled individually with a joystick via torque control. Cartesian control allows control of the arm in open loop using inverse kinematics, thus allowing the user to move the arm in a global X, Y Z direction.The final control mode, joint angle control, consists of a closed-loop control of the arm to perform a pre-programmed trajectory path. The controller for this node is illustrated in the control diagram for the fixed base configuration in figure 10.2. Diagrams 10.6 through 10.8 show the overall structure of the Arm Controller Node and the 3 control modes.

With this framework in place, a complete simulated refueling task was per-

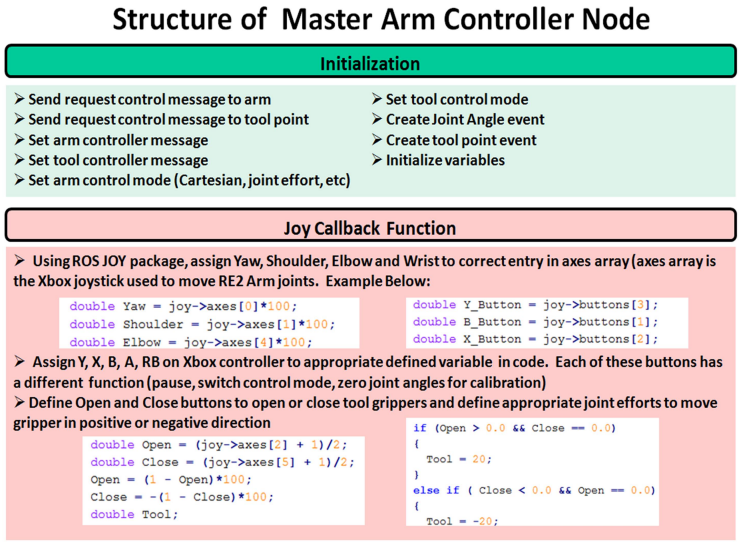

Figure 10.6. Overview of master control node initialization and callback functions.

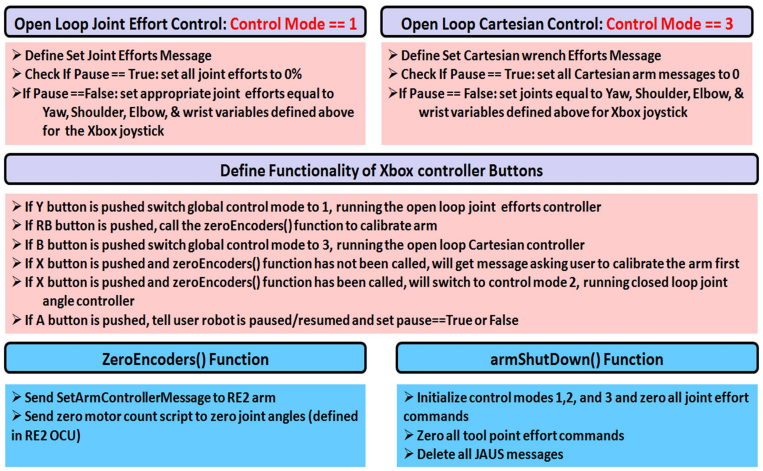

Figure 10.7. Open loop cartesian and joint control modes for RE2 arm.

formed using the RE2 arm in the closed loop control mode, labeled as control mode 2 in Figure 10.8. This closed loop control mode allowed the RE2 arm to perform a semi-autonomous refueling task, which is covered in the next section.

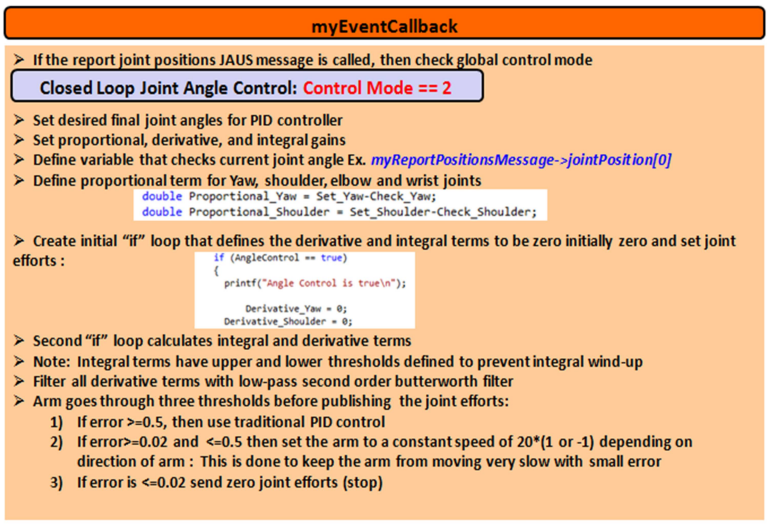

Figure 10.8. Closed-loop joint angle control algorithm structure.

10.5 Test Setup

The geometry of several gas tanks was compiled in Table 10.1 for several representative fueling tasks, based on common fuel tank inlet locations for military vehicles and generators. This table looks at the angle of the filler neck, the height, and inner diameter of the fuel filler tube attached to a vehicle gas tank. Figure 10.9 shows gas tank configurations for the HEMTT A2 above and HEMTT A4.

Table 10.1. Table of gas tank geometries for several vehicles.

Vehicle Make and Model	Height of Fuel Neck [in]	Inner Diameter [in]	Approx. Fuel Neck Angle from Horizontal
HEMTT A4	51	4.75	65
HEMTT A2	51	4.75	65
M915	35	4.75	65

Using the approximate geometry from sampled gas tanks, a general refueling test platform was created to test nozzle entry into fuel tanks at several different angles. Additionally, a CAD model was developed to conduct the same tests within the GAZEBO environment. A SolidWorks model of the refueling platform is shown in Figure 10.10.

A testing room was created for the robot to conduct simulated refueling tasks

Figure 10.9. Truck and corresponding gas tank configurations for the HMTT A2 (above) and HMTT A4 (below).

with the RE2 arm. Using the measured tank geometry, the refueling platform is mounted onto a peg-board wall for testing. The height of the platform and the global location with respect to the RE2 arm are both adjustable. Figure 10.11 illustrates this setup.

10.6 Fixed Base

10.6.1 Physical Testing of a 90 degree Refueling Neck Orientation

The first test was conducted for a simple refueling orientation of 90 degrees. The path of the arm end effector follows a straight line path until it aligns with the vertical axis of the gas tank. The arm then travels down vertically along to global Z axis until it aligns with the gas tank opening. This path can be seen in Figures 10.12 and 10.13.

The plots in Figures 10.12 through 10.18 show the data collected for the 90 degree refueling neck orientation and a diagram of the test setup. It can be seen in the trajectory solution for this path that there is a point of instantaneous change

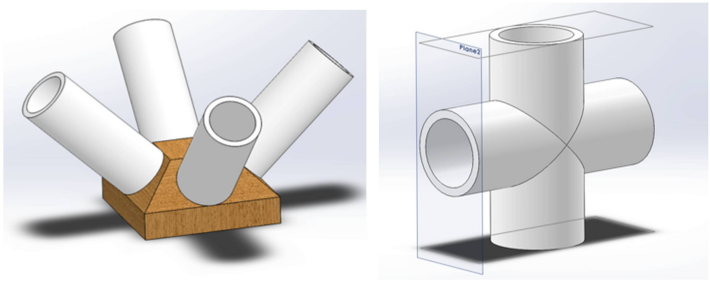

Figure 10.10. SolidWorks model of refueling platform (left) and additional 0 and 90 degree tank configurations (Right).

in slope for the joint angles. This causes oscillation in the solution as the derivative term and Newton Raphson solution tries to instantaneously adapt to this point. This can be prevented in the future by smoothing out sharp movements in the trajectory; methods for smoothing such trajectories are found in R. Jazars textbook [18]. A plot comparing the reference trajectory and followed trajectory, as well as the Proportional, Integral and Derivative terms of the PID controller for each joint is shown in Figures 10.13 through 10.18.

10.6.2 Testing for a 45 degree Refueling Neck Orientation

Physical testing of a 45 degree fuel port configuration was conducted with the inclusion of a wrist end effector for more precise orientation of the refueling tube. Figures 10.19 through 10.21 are pictures of the arm navigating to the refueling platform, and aligning along the tank center axis during testing

After completing physical tests of a refueling task for both vertical and 45 degree fuel inlet orientations, it was determined that an additional wrist joint was needed to properly align a nozzle with the central axis of the fuel inlet. Although the kinematic equations derived in Chapter 5 allowed a 3 degree of freedom arm to follow a desired trajectory path, an additional joint is needed to define the pose

Figure 10.11. Test platform setup.

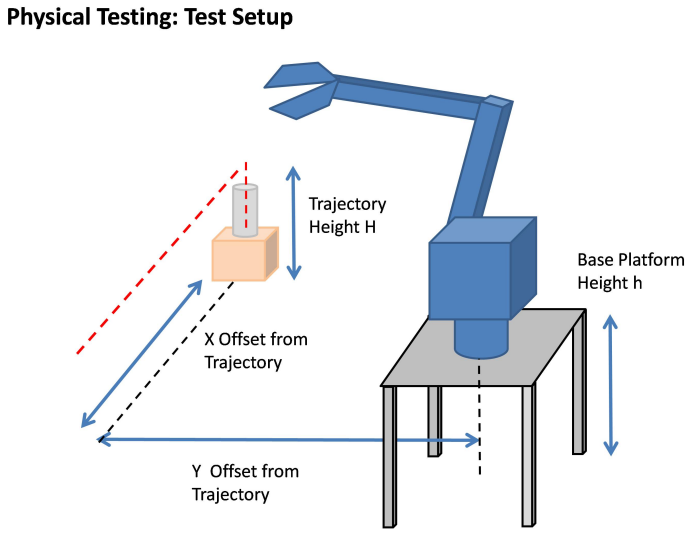

Figure 10.12. Diagram of 90 degree refueling orientation.

and orientation of a trajectory point as it travels along the central axis of a 45 of 65 degree fuel tank inlet. As seen in Figures 10.19 through 10.21, a wrist joint was used for physical testing of the 45 degree fuel inlet to ensure the nozzle was aligned with the central axis during the task. The wrist joint attachment on the RE2 arm is open loop, meaning there is no angular feedback on the wrist joint that could

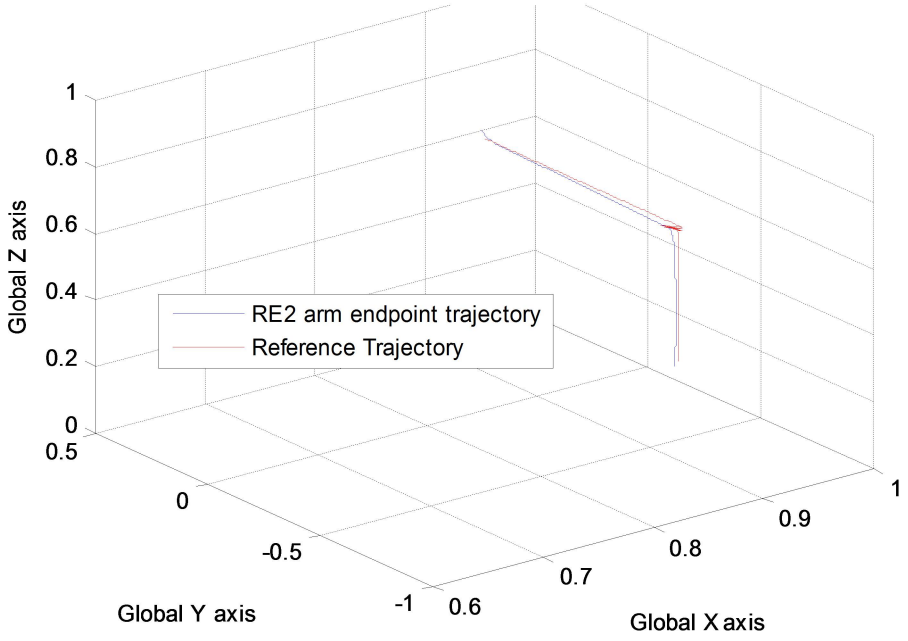

Figure 10.13. Plot of reference and actual endpoint trajectory. Units in Meters [m].

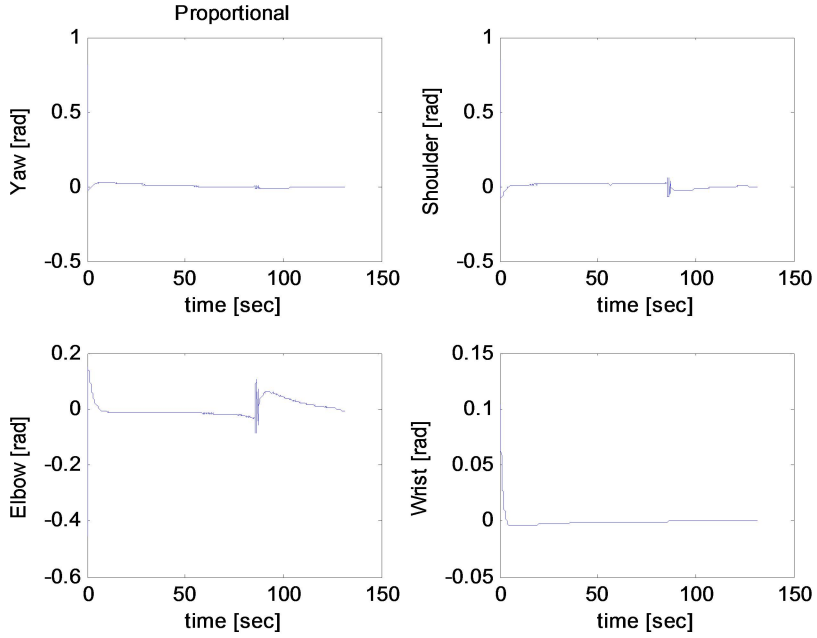

Figure 10.14. Proportional terms for RE2 arm joint angles for a refueling task.

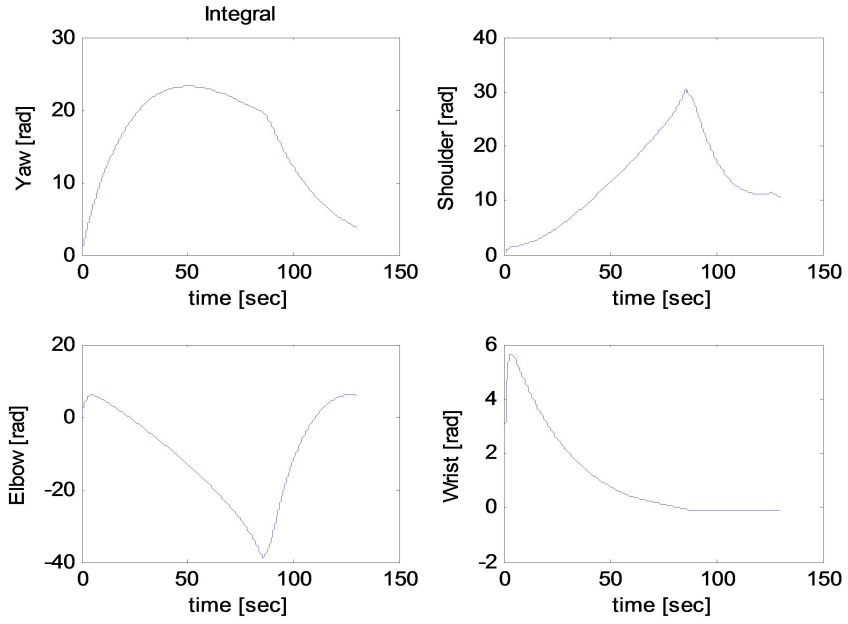

Figure 10.15. Integral terms for RE2 arm joint angles for a refueling task.

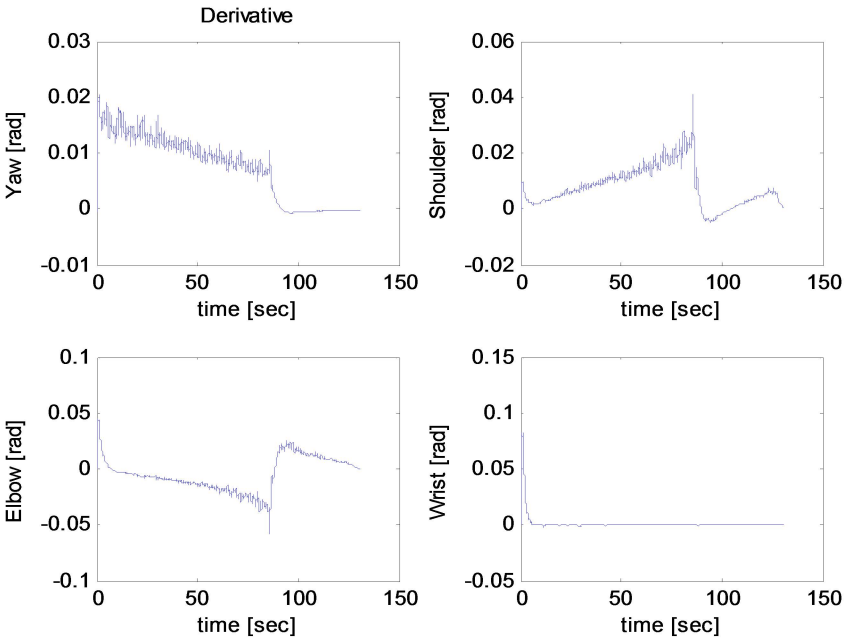

Figure 10.16. Derivative terms for RE2 arm joint angles for a refueling task.

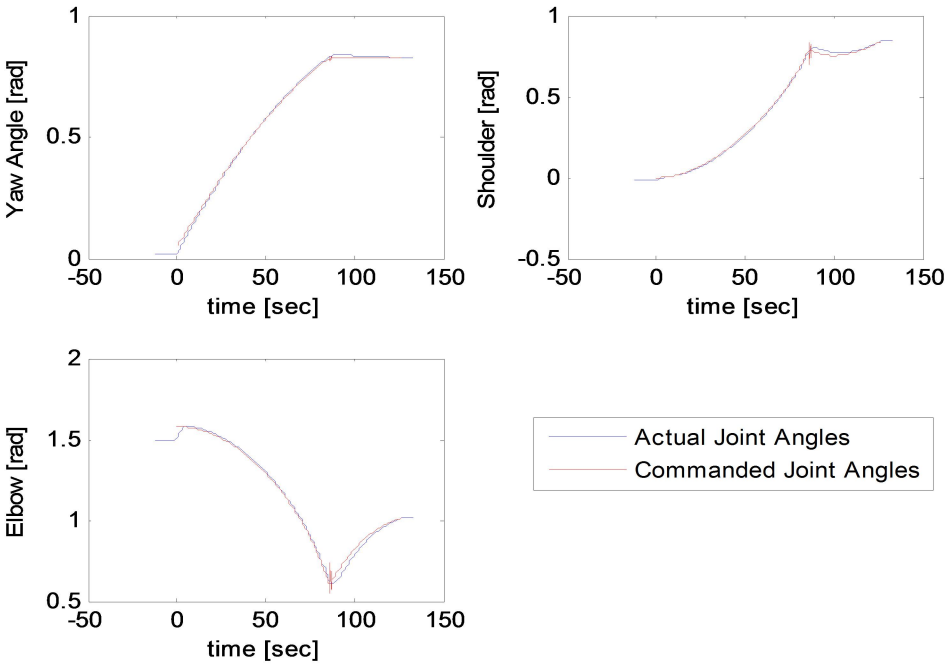

Figure 10.17. Plot of reference and actual trajectories of each joint for a refueling task.

Figure 10.18. End configuration of RE2 arm for 90 degree refueling task.

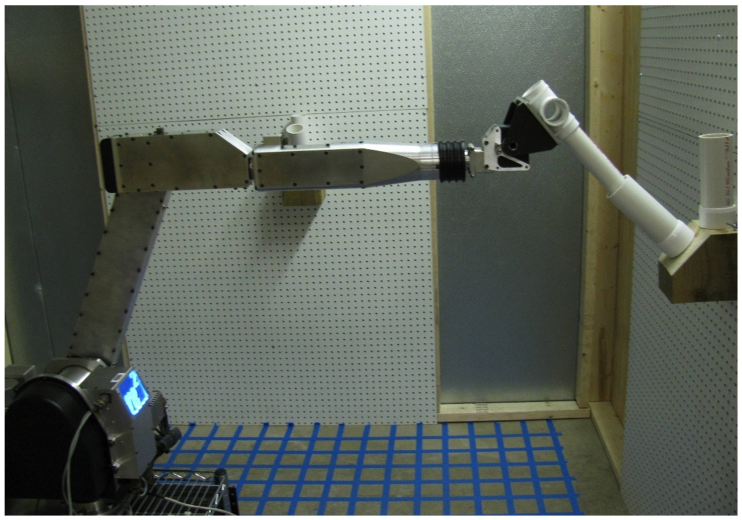

Figure 10.19. Snapshots of the RE2 arm as it aligns along the tank fuel port axis during testing of 45 degree orientation.

be used within the control system shown in Figure 10.3. The next step would be to incorporate a wrist joint with feedback to be used for a refueling task. Through the incorporation of a wrist joint with feedback, the RE2 arm could successfully complete a refueling task for 45 and 65 degree fuel inlets. A camera could also be placed on the end effector to fine tune alignment of the wrist joint as it performs the refueling task, allowing for more autonomy within the robotic system.

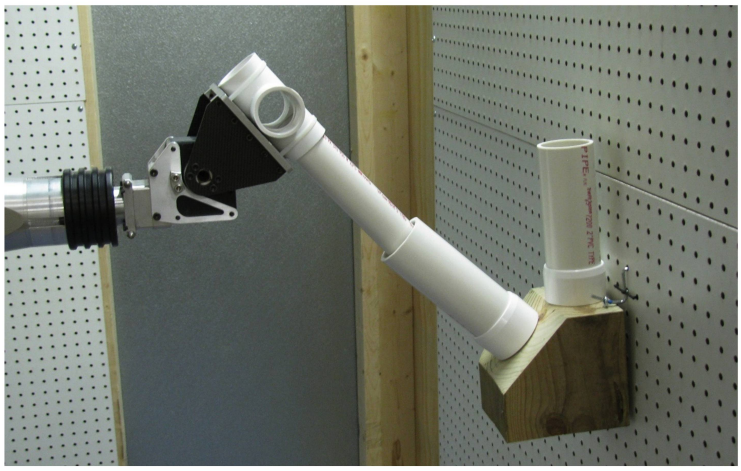

Figure 10.20. Illustration of Nozzle Alignment.

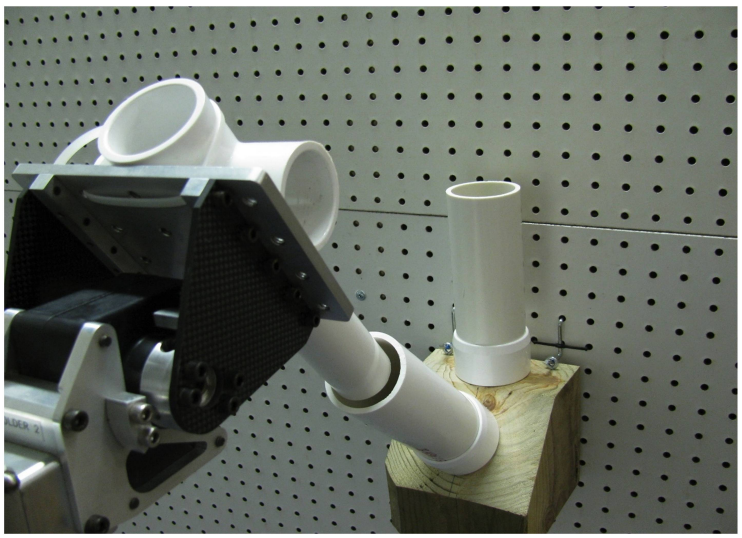

Figure 10.21. Close up of Nozzle Alignment.

Chapter 11

Conclusions

11.1 Testing of Simulated Refueling Trajectory

This thesis looked at the ability of a robotic arm to follow a refueling trajectory that simulated nozzle insertion into a fuel tank inlet. This analysis was first verified through simulation, and then tested with a three-degree-of-freedom RE2 Automatic Arm. The RE2 arm was fully incorporated with ROS (Robot Operating System) and GAZEBO, a 3D simulation package, to test refueling trajectories. The geometry of several vehicle fuel tanks was recorded. Sample refueling trajectories were then created that simulated nozzle insertion into fuel tank inlets at a 90, and 45 degree angles along the inlet central axis.

The three-degree-of-freedom RE2 robotic manipulator followed a 90 degree trajectory path, where it successfully inserted a nozzle along a vertical central axis of a modeled gas tank inlet. The success of the nozzle insertion was dependent on mounting the nozzle at a convenient orientation on the robotic end effector that would properly align with the fuel tank inlet. When performing a second simulated trajectory with a 45 degree insertion angle, a wrist joint was mounted to the RE2 end effector that added a fourth degree-of-freedom. This wrist joint was necessary to successfully align the nozzle along the central axis of the 45 degree fuel inlet, and would be required to accomedate other alignment angles. The testing of the RE2 arm to simulate nozzle insertion into the fuel tank was conducted with a fixed base robotic manipulator. The fixed base analysis for a refueling task was next extended to incorporate mobile base movement.

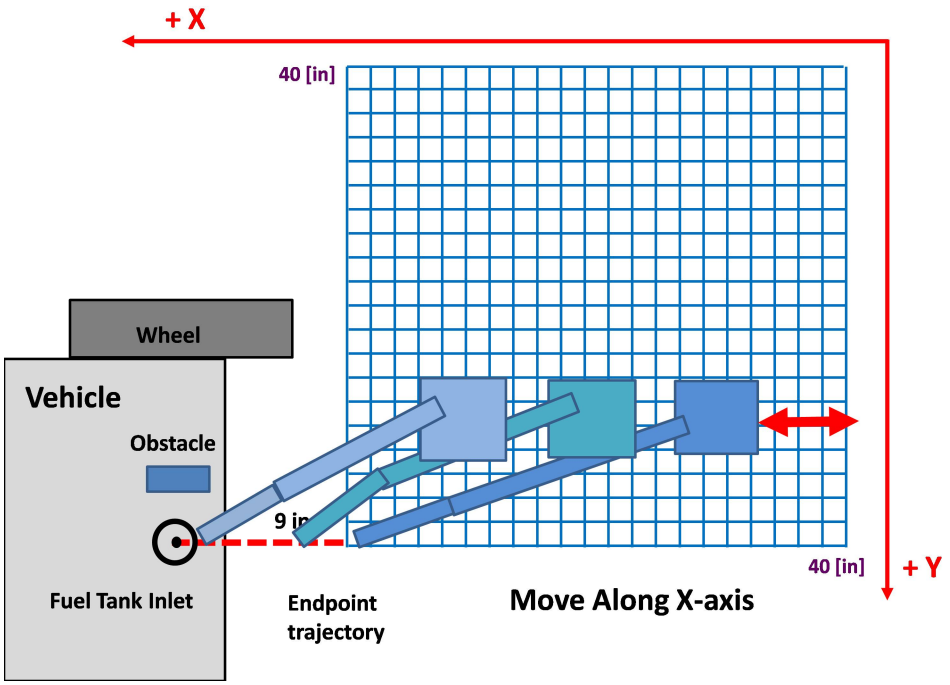

Figure 11.1. Illustration of mobile base movement along the Y axis with a fixed X offset while the end effector follows a commanded trajectory.

11.2 Benefits of Manipulator and Base Coordination

The goal of using a mobile base with a robotic arm was to investigate whether coordination between mobile base movement and arm movement could benefit the robustness of a refueling task. It was found that when the robotic manipulator was allowed to move perpendicular to the fuel inlet of the gas tank, it was necessary for the mobile base to move closer to the gas tank as the end effector reached the end of the trajectory path. This movement is illustrated again in Figure 11.1. By allowing the base to move during the refueling task, the robotic manipulator had the ability to adjust its base location to maintain optimization metrics such as minimizing joint velocities or power consumption as the robotic manipulator traced the trajectory path. The mobile manipulator can also adjust its base location to avoid singularities, collisions or any obstructions that may lie around the fuel tank inlet.

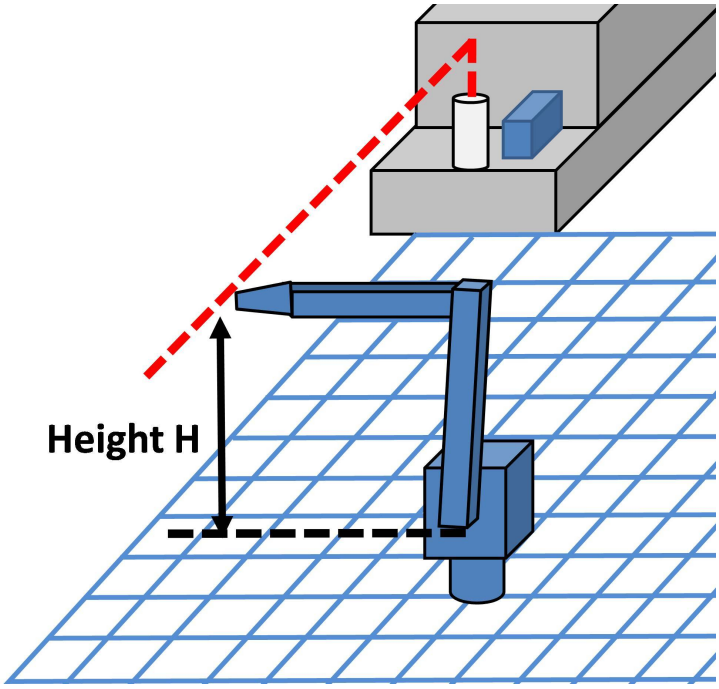

Figure 11.2. Figure of trajectory height that is similar to the length of the shoulder linkage.

11.3 Optimal Trajectory Height for Mobile Base Design

In addition to utilizing mobile base movement to facilitate a refueling task, the height of the trajectory in relation to the manipulator origin was also investigated. The cost plots for varying trajectory heights are shown in Chapter 8. These cost plots illustrate that larger heights limit the area where the mobile robot can be in relation to the vehicle fuel tank to conduct a successful refueling task. The total length of the RE2 Arm is roughly 52 inches. With a trajectory height of 42 inches relative to the origin of the robotic arm, the amount of available space for the manipulator base to position to for a successful refueling task is greatly reduced. Higher trajectory heights lead to more manipulator kinematic lockup for base locations that are further away from the fuel tank. This is because certain trajectory points may lie outside or on the edge of the arm's workspace.

When the height of the trajectory was reduced, it was found that lower heights relative to the manipulator origin greatly widened the optimal low-cost basin.

In addition, it was found that a trajectory height of 18 inches relative to the manipulator origin was optimal. The trajectory height of 18 inches was similar to the shoulder linkage length of 20 inches. By keeping the trajectory height close to the height of the shoulder link, the elbow joint moved at a minimal speed because the elbow joint itself was only required to make small adjustments to the height of the end effector tip. The shoulder joint speed was also minimized because the shoulder joint was only needed to extend out the arm for trajectory points that were further away from the arm. For trajectory points closer to the arm, only minimal shoulder joint movement was needed because the height of the trajectory was already level with the elbow joint. The yaw joint slowly rotated the arm to position the end effector along the trajectory path. Overall, this configuration reduced the maximum joint speeds and created a wider space of available base locations for the mobile robot. Figure 11.2 illustrates this configuration. A picture of the optimal trajectory height of 18 inches relative to the origin of the robotic arm is shown again in Figure 11.3. Further study should be conducted to adjust linkage lengths of the arm to illustrate how these linkage lengths affect the amount of reachable space for a specific refueling task.

An optimal trajectory height must also be compared to the fuel inlet heights of several vehicles listed in Table 10.1 in Chapter 10. Because the fuel tank inlet height for several vehicles is a few feet off the ground, it makes more sense for the arm to reach up for a refueling task rather than down. Reaching down would imply that the mobile base is several feet off the ground, which could introduce stability issues to prevent tipping, and could also cause more collisions with obstacles located near the fuel tank inlet and above. Because there may be different fuel inlet heights on different vehicles, a best height may be chosen that maximizes the available space across multiple vehicles. This means that the mobile base height will be designed to accommodate a wide range of different vehicles for a refueling task.

In addition to the design of mobile base height, the combined cost plots can provide further information on the tolerances needed for robot global position estimation and wheel feedback. These results provide guidance on the accuracy of the sensors and control system used to position the mobile base next to the fuel tank. If the robot lies in the center of the optimal basin, error within +/- 6 inches

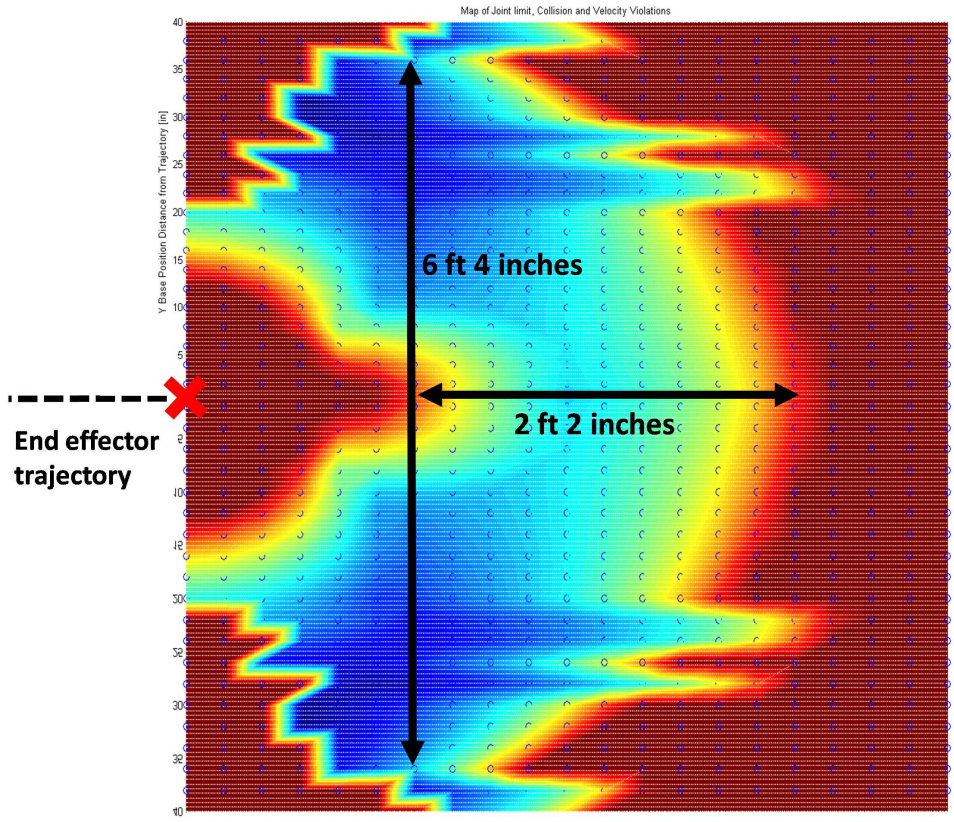

Figure 11.3. Aerial view of a 3D cost plot combining percent of maximum joint velocities, singularities, collision, and joint limit violations for a 90 degree reference trajectory height of 18 inches.

may not be an issue for the robot. If the basin were smaller, which may occur if the height between the manipulator origin and the trajectory were larger, these base positioning error tolerances would need to be much tighter.

11.4 Applications

Because the robot has the ability to reference pre-calculated optimal arm insertion trajectories both on-line and off-line for a specific refueling task, the robot can utilize motion of the base (by driving the wheels or tracks) to minimize certain joint movements. This information will be useful in the field specifically for scenarios where movement in a certain joint may be limited. For example, if the motor for the shoulder joint on a robot is running below a desired level of performance, the robot can adjust its base location throughout the trajectory to ensure the

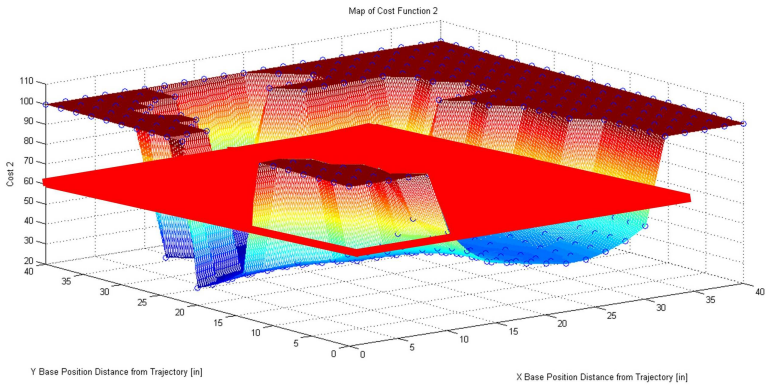

Figure 11.4. Figure of total cost threshold that can be established for cost functions, enabling the robot to only consider base positions that have costs below a certain value.

shoulder joint does not go over a certain percent of maximum velocity for that particular motor. Figure 11.4 shows the cost map for the second cost function. The second cost function penalized the shoulder joint movement throughout the entire trajectory. As a result, areas of low cost illustrate base locations where shoulder joint movement is minimal. Drawing an example from this cost function, if the robot knew the shoulder joint could only operate at a certain percent of maximum velocity value, the robot could eliminate all costs that lie above a certain value, and only consider base locations that fell at or below this desired cutoff. The robot could then simply choose one of the base locations closest to its current position.

11.5 Future Work

When considering mobile base movement in this thesis, the mobile base motion was considered only along the global X or Y axis with a fixed Y or X offset relative to the fuel tank. Referencing Figure 11.5, the mobile base can also be considered to optimize its location at a diagonal path, to utilize the widest geometry of the low cost basin. Using the 3D cost plots, the next step would be to apply the same methodology used in this thesis, to base motion that followed the widest geometries of the low cost region, allowing the base to move at a curved or diagonal path, rather than simply along the X, or Y axis.

Throughout this thesis, many subtasks such as removal of the fuel cap, positioning of the robot, etc. were assumed to be completed successfully prior to the

trajectory control of the refueling nozzle. Additional trade studies are required to characterize the requirements for sensing the position and relative orientation of the fuel filler tube with respect to the coordinate system of the refueling system. Possible sensing approaches include the use of LIDAR, stereo vision, RADAR, or ultrasonic sensors to scan the vehicle/equipment and identify the shape and location of the filler tube (and cap). An alternative approach would be to outfit the filler tube/cap with a distinguishing and non-ambiguous fiducual that could be easily recognized and localized. One potential technology would be an RFID sensor, which could also provide vehicle identification information to automate the transactional side of the refueling option as well as the fuel delivery.

A final consideration in the design of an automated refueling system is subsystem and component characterization and testing. Specifically, developments of standardized test methods are needed to characterize and quantify performance of simple and complex tasks such as inspection, mapping, mobility, and manipulation. None of these by themselves match the complexity of the automated refueling task, but individually they capture the requirements of the key subsystems and can be used to characterize the mobile base, manipulator arm, sensors, and end effectors.

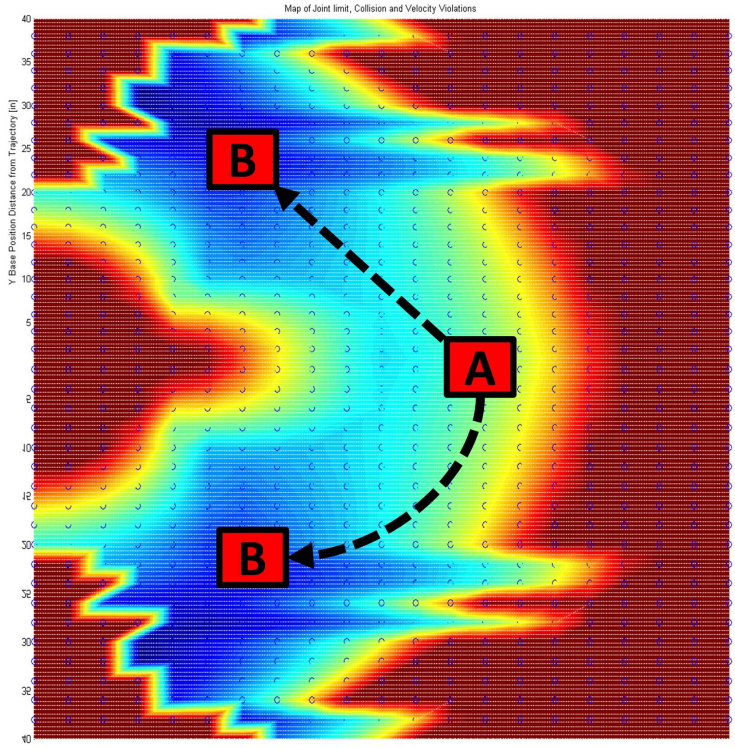

Figure 11.5. Illustration of future mobile base paths to be explored for a mobile robot carrying out a refueling task

Appendix A

Matlab and ROS C++ Code

This Appendix section provides sample code used throughout the thesis to implement refueling simulations via MATLAB and GAZEBO.

A.1 ROS Nodes

A.1.1 Joint Publisher

The C++ code below consists of a joint publisher node that publishes joint angles to the GAZEBO simulation via predetermined topics. This node publishes joint angles to the appropriate topic that GAZEBO, or the RE2 arm subscribes to.

```
#include "ros/ros.h"
#include <iostream>
#include "std_msgs/Float64.h"
#include <cmath>
#include <vector>
#include <sstream>
#include "std_msgs/Header.h"

int i =0;

int main(int argc, char **argv)
{
```

```
ros::init (argc, argv, "joint_publisher");
ros::NodeHandle nh;

ros::Publisher Joint1_pub =
nh.advertise<std_msgs::Float64>
("/re2/joint1_position_controller/command",1);
ros::Publisher Joint2_pub =
nh.advertise<std_msgs::Float64>
("/re2/joint2_position_controller/command",1);
ros::Publisher Joint3_pub =
nh.advertise<std_msgs::Float64>
("/re2/joint3_position_controller/command",1);
ros::Publisher Joint4_pub =
nh.advertise<std_msgs::Float64>
("/re2/joint4_position_controller/command",1);
ros::Publisher Time_pub =
nh.advertise<std_msgs::Header>
("/re2/time",1);

ros::Rate loop_rate(20);

while (nh.ok ())
{
  std_msgs::Float64 Joint1Msg;
  std_msgs::Float64 Joint2Msg;
  std_msgs::Float64 Joint3Msg;
  std_msgs::Float64 Joint4Msg;
  std_msgs::Header TimeMsg;

  //Set the desired joint angle

  double zero_time =
       ros::Time::now().toSec();
```

```
double check_time =
    ros::Time::now().toSec();

while (check_time < zero_time+3)
{
  double Set_Yaw = 0;
  double Set_Shoulder = 0;
  double Set_Elbow = 0;
  double Set_Wrist = 0;

  Joint1Msg.data = Set_Yaw;
  Joint1_pub.publish(Joint1Msg);

  Joint2Msg.data = Set_Shoulder;
  Joint2_pub.publish(Joint2Msg);

  Joint3Msg.data = Set_Elbow;
  Joint3_pub.publish(Joint3Msg);

  Joint4Msg.data = Set_Wrist;
  Joint4_pub.publish(Joint4Msg);

  check_time =
      ros::Time::now().toSec();

  TimeMsg.stamp = ros::Time::now();
  Time_pub.publish(TimeMsg);
}

zero_time = ros::Time::now().toSec();
check_time = ros::Time::now().toSec();

while (check_time < zero_time+3)
```

```
    {
      double Set_Yaw = 0.5;
      double Set_Shoulder = 0;
      double Set_Elbow = 0;
      double Set_Wrist = 0;

      Joint1Msg.data = Set_Yaw;
      Joint1_pub.publish(Joint1Msg);

      Joint2Msg.data = Set_Shoulder;
      Joint2_pub.publish(Joint2Msg);

      Joint3Msg.data = Set_Elbow;
      Joint3_pub.publish(Joint3Msg);

      Joint4Msg.data = Set_Wrist;
      Joint4_pub.publish(Joint4Msg);

      check_time =
          ros::Time::now().toSec();

      TimeMsg.stamp = ros::Time::now();
      Time_pub.publish(TimeMsg);
    }
    i=i+1;
    loop_rate.sleep();
  }

  return 0;
}
```

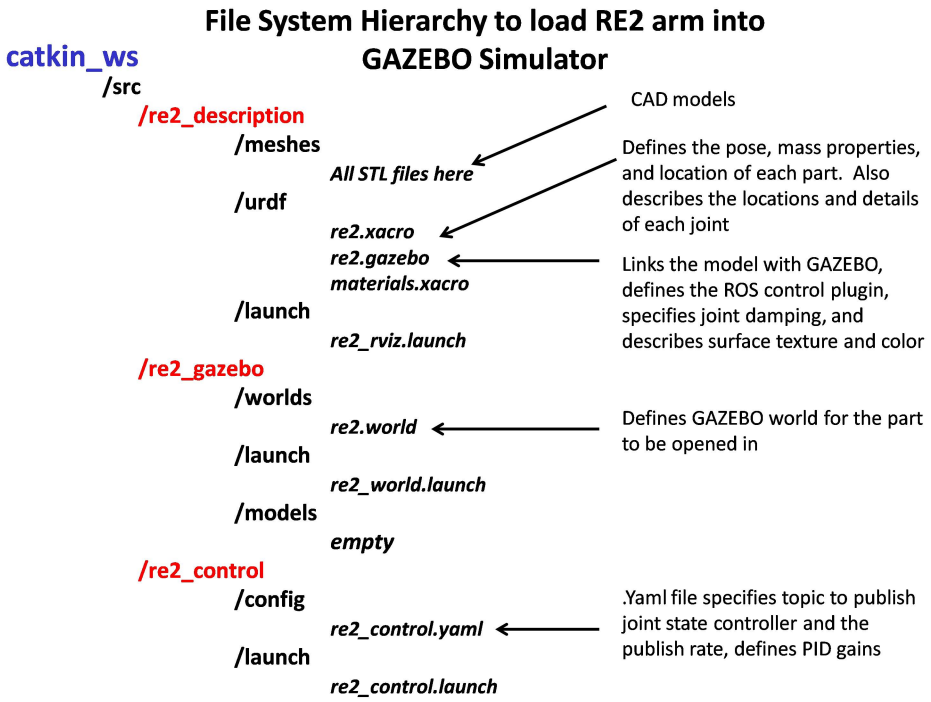

Figure A.1. File structure within ROS used for the GAZEBO refueling simulations

A.1.2 ROS-GAZEBO File Structure

Figure A.1 illustrates the file structure within GAZEBO used for the RE2 arm simulations.

A.2 Matlab Scripts

A.2.1 Newton-Raphson Inverse Kinematics

```
numbers
%Create the Jacobian for RE2 arm
%Set dimensions of RE2 arm
set(0,'DefaultFigureWindowStyle','docked')
%Arm geometries, dimensions in meters
a1 = 0.0762;
a2 = 0.52705;
a3 = 0.00635;
a4 = 0;
d1 = 0.2921;
d0 = 0;
d3 = 0;
```

```
d2 = 0.0381;
d4 = 0.2413;
d5 = 0.5461;       %0.6858;
residual=1E6;
thresh=1E-4;
violation2=0;
violation3=0;

% Kinematic Equations for Position and Velocity
% This script illustrates the inverse kinematics used to find joint angles,
% and joint velocities for a given trajectory coordinate and velocity

q_guess(1,1)=theta1;
q_guess(2,1)=theta2;
q_guess(3,1)=theta3;
q_first_guess=q_guess;
count=0;
iteration=1;
loop_count=0;

%X and Y-offsets fed into function are varied along the grid, initially a
%Y-axis offset is fed into the function, where an outer loop varies the
%X-offset values
for l=1:1:loop_number %Varies X-offset
    %This was used to also change Y-offset.  This is set to a value of 1
    %allowing a separate function to determine the Y-offset
    k=1;%for k=1:1:1
    %q_guess provides initial an joint angle guess for the Newton-Raphson
    %solution
    q_guess=[q_first_guess(1,1); q_first_guess(2,1); q_first_guess(3,1);];
    theta1=q_guess(1,1);
    theta2=q_guess(2,1);
    theta3=q_guess(3,1);

    for i=1:1:251
        count=count+1;
        %Define end effector trajectory
        if i<201
        %Computes trajectory coordinated based upon X and Y offsets
        %Straight line portion of the trajectory
           Px=((2*(l-1))*0.0254)+((9/200)*i*0.0254);
           Py=((y+(2*(k-1)))*0.0254);
           Pz=Z_init;
           x_dot=[-0.1; 0; 0;];
         end

         if i>201
            %Vertical portion of the trajectory
            if (i-200)<0
                iz=(i-200)*1;
```

```
            else
                iz=(i-200);
            end
            Px=((2*(l-1))*0.0254)+((9/200)*200*0.0254);
            Py=((y+(2*(k-1)))*0.0254);
            Pz=Z_init-((iz/50)*4*0.0254);
            x_dot=[-0.1; 0; -0.1;];
        end
        %Check PY,PY,PZ used for debugging
        check_PX(i,1,k,l)=Px;
        check_PY(i,1,k,l)=Py;
        check_PZ(i,1,k,l)=Pz;

        while residual>1E-4 %This tolerance is set for the residual term
              %Plug in joint angles and geometries to find the Jacobian
              [X,Y,Z,J]=Jacobian(theta1,theta2,theta3,a1,a2,a3,a4,d1,d2,...
                  d3,d4,d5);
              %Calculate the inverse Jacobian
              inv_Jac=inv(J);
              residual=max(abs([Px; Py; Pz;]-[X; Y; Z;]));
              %Newton Raphson Method
              q=q_guess+(inv_Jac*([Px; Py; Pz;]-[X; Y; Z;]));
              theta1=q(1,1);
              theta2=q(2,1);
              theta3=q(3,1);
              %Unwrap all joint angles
              [theta1,theta2,theta3]=unwrap(theta1,theta2,theta3);
              %set guess matrix
              q(1,1)=theta1;
              q(2,1)=theta2;
              q(3,1)=theta3;
              %Stores previous pose for a guess of next trajectory point
              q_guess=q;
              loop_count=loop_count+1;
              %This allows user to break out of loop if solution
              %diverges
              if loop_count >= 100
%                 msg=sprintf('diverging');
%                 fprintf(msg)
                  break
              end
        end             %Stores previous pose to use as
        if i==1
            q_first_guess=q_guess;
        end
        %Check X, Y, Z and residual values for debugging
        loop_count=0;
        check_X(i,1,k,l)=X;
        check_Y(i,1,k,l)=Y;
        check_Z(i,1,k,l)=Z;
```

```
%Store theta values
theta_1(i,1,k,l)=theta1;
theta_2(i,1,k,l)=theta2;
theta_3(i,1,k,l)=theta3;
theta_4(i,1,k,l)=0;
%Set residual to large value for next trajectory point
residual=1E6;
% Calculate Joint Velocities
%Recaluculate the Jacobian, and inverse Jacobian based on final
%joint angle values
[J,inv_Jac]=inverseJacobian(theta1,theta2,theta3,a1,a2,a3,a4,d1,...
    d2,d3,d4,d5);
% Solve for angular velocity
q_dot=inv_Jac*x_dot;
theta1_dot(i,1,k,l)=q_dot(1,1);
theta2_dot(i,1,k,l)=q_dot(2,1);
theta3_dot(i,1,k,l)=q_dot(3,1);

%Check Forward Kinematics
% Denavit-Hartenberg parameters
DH=[0   0    d1 theta_1(i,1,k,l); -a1 -pi/2 d2  theta_2(i,1,k,l);...
    a2  0    0 theta_3(i,1,k,l); a3 -pi/2 d4 theta_4(i,1,k,l);];
%Find transformation matrices for each joint
[T_01,T_12,T_23,T_34,T_45]=ForwardKinematics(DH,d5);

T_05 = T_01*T_12*T_23*T_34*T_45;    %Track X,Y,Z endpont P
T_03 = T_01*T_12*T_23;   %Track coordinates of elbow joint
T_02 = T_01*T_12;
%Store all joint coordinates and plug all angles back into
%the forward kinematic equations to verify corectness
X5(i,1,k,l)=T_05(1,4);
Y5(i,1,k,l)=T_05(2,4);
Z5(i,1,k,l)=T_05(3,4);

X3(i,1,k,l)=T_03(1,4);
Y3(i,1,k,l)=T_03(2,4);
Z3(i,1,k,l)=T_03(3,4);

X2(i,1,k,l)=T_02(1,4);
Y2(i,1,k,l)=T_02(2,4);
Z2(i,1,k,l)=T_02(3,4);

X2_1(i,1,k,l)=T_02(1,4);
Y2_1(i,1,k,l)=T_02(2,4);
Z2_1(i,1,k,l)=T_02(3,4);

X1(i,1,k,l)=T_01(1,4);
Y1(i,1,k,l)=T_01(2,4);
Z1(i,1,k,l)=T_01(3,4);
```

```
            X0(i,1,k,l)=0;
            Y0(i,1,k,l)=0;
            Z0(i,1,k,l)=0;
        end
end
```

A.2.2 Unwrap Function

```
numbers
function [theta1,theta2,theta3]=unwrap(theta1,theta2,theta3)
%This function unwraps joint angles to range from -pi to pi
while theta1>pi
      theta1=theta1-2*pi;
end

while theta1<-pi
      theta1=theta1+2*pi;
end

while theta2>pi
      theta2=theta2-2*pi;
end

while theta2<-pi
      theta2=theta2+2*pi;
end

while theta3>pi
      theta3=theta3-2*pi;
end

while theta3<-pi
      theta3=theta3+2*pi;
end

end
```

A.3 Ray Casting

Portions of ray casting script are from HJ Sommer's notes and example script on ray casting with imported CAD files [2].

```
    numbers
function [collision,r_ray,d_ray,F2,V2,C2,len,r_end]=...
    collision_check(T03,T05,X_position,y,Z_init)
% read_stl_binary.m - read binary STL file and display
```

```
% KLB Modified Code from:HJSIII, 14.05.22
%This function is called to perform a collision check
%as the arm moves along a trajectory.
%The coordinates of the arm joints are passed into this function,
%and the collision function checks if there is a collision
%between these coordinates

% file name
f_name = 'obstruction_block.STL';

% open file
f_id = fopen( f_name, 'r');
if f_id == -1
  error( 'File could not be opened' )
end

% read 80 character header
header = fread( f_id, 80, 'uchar' );
%disp( char( header' ) )

% read number of facets
nfacet = fread( f_id, 1, 'int32' );
%disp( [ num2str(nfacet) ' facets' ] )

% read normals and vertices - skip two bytes at end of each
% nx,ny,nz, x1,y1,z1, x2,y2,z2, x3,y3,z3, padding
for i = 1:nfacet,
  tri(i,:) = fread( f_id, 12, 'float32' );
  padding = fread( f_id, 2, 'uchar' );
end

% close file
fclose( f_id );

% (X,Y,Z,C) format for patch

% create arrays for patch - each is 3 x nfacet
X = [ tri(:,4)'  ;
      tri(:,7)'  ;
      tri(:,10)' ];

Y = [ tri(:,5)'  ;
      tri(:,8)'  ;
      tri(:,11)' ];

Z = [ tri(:,6)'  ;
      tri(:,9)'  ;
      tri(:,12)' ];

% pink color
```

```
C(:,:,1) = ones( size(X) );
C(:,:,2) = 0.9*C(:,:,1);
C(:,:,3) = 0.9*C(:,:,1);

% append all vertices together - many will be repeated
nver = nfacet*3;
V = zeros( nver,3 );
%  allocate space to save time
V = [ tri(:,4:6)   ;
      tri(:,7:9)   ;
      tri(:,10:12) ];

%Replace small numbers with zero
for i=1:1:3
    for j=1:length(V)
        if V(j,i) <= 1E-4
            V(j,i)=0;
        end
    end
end
% facet list
f1 = ( 1 : nfacet )';
f2 = f1 + nfacet;
f3 = f2 + nfacet;
F = [ f1  f2  f3 ];  % nfacet x 3

% color
C = ones(nver,1) * [ 1.0  0.9  0.9 ];

% prune vertex list

% initialize look-up table to convert old indices to new
index = ( 1 : nver )';

% start new vertex list
nver2 = 1;
V2 = V(1,:);

% check each old vertex
eps = 0.001;
for i = 2:nver,
  vo = V(i,:);

  % check for match with vertices
  t = ( vo(1)==V2(:,1))+(vo(2)==V2(:,2))+(vo(3)==V2(:,3));
  [ max_t, j ] = max(t);

  % append to new list if no match, or record which one it does match
  if max_t<3,
    nver2 = nver2 + 1;
```

```
    V2(nver2,:) = vo;
    index(i) = nver2;
  else
    index(i) = j;
  end
end

% With local coordinates of V2 provided in STL file,
% will need to translate points to correct Global location,
% providing a description of the vertices with respect
% to the global coordinate frame
V2(:,1)= (V2(:,1)*0.0254)+((2*(X_position-1)*0.0254)...
    +(9*0.0254))-(2*0.0254);
V2(:,2)= (V2(:,2)*0.0254)+((y*0.0254)-(4*0.0254));
V2(:,3)= (V2(:,3)*0.0254)+(Z_init-(4*0.0254));

% update face list
F2 = index( F );

% color
C2 = ones(nver2,1) * [ 1.0  0.9  0.9 ];

% ray casting - finite length ray
r_ray = T03; %location of origin of last  link (T_03)
ray_calc=T05-T03;
u_hat = ray_calc/norm(ray_calc); % T06-T03/normalized value
%Length should be length of last two links, from frame 3 to point P,
%Coordinates are used to comupute distance formula between P and 3
%This distance is used to check for collisions.
%The last two links occur after the elbow joint of the arm
len = sqrt(power((T03(1)-T05(1)),2)+power((T03(2)-T05(2)),2)+...
    power((T03(3)-T05(3)),2));
d_ray = len * u_hat;
r_end = r_ray + len*u_hat;
%quiver3( r_ray(1),r_ray(2),r_ray(3), d_ray(1),d_ray(2),d_ray(3), 'r' )

% check if ray pierces each facet
for ifacet = 1 : nfacet,
  p1 = V2( F2(ifacet,1), : )';
  p2 = V2( F2(ifacet,2), : )';
  p3 = V2( F2(ifacet,3), : )';

% form rotation matrix
% p1 at local origin
% x local along p1 to p2
% y local perpendicular to p1 to p2 in direction of p1 to p3
% z local perpendicular to plane of p1, p2, p3
  f_hat = p2 - p1;
  f_hat = f_hat / norm(f_hat);
```

```
  h_hat = cross( f_hat, (p3-p1) );
  h_hat = h_hat / norm(h_hat);

  g_hat = cross( h_hat, f_hat );
  A = [ f_hat  g_hat  h_hat ];

% find intersection with extended plane
  n_hat = h_hat;
  s = n_hat' * (p1 - r_ray) / (n_hat' * u_hat);
  p_ray = r_ray + s * u_hat;

% local x-y coordinates -  all must have local z = 0
  loc1 = A' * (p1-p1);
  loc2 = A' * (p2-p1);
  loc3 = A' * (p3-p1);
  loc_ray = A' * (p_ray-p1);

% test if piercing point is inside facet
  xv = [ loc1(1)  loc2(1)  loc3(1) ];
  yv = [ loc1(2)  loc2(2)  loc3(2) ];
  in = inpolygon( loc_ray(1), loc_ray(2), xv, yv);
  if in==1,

% test for ray direction and length
  if (s>=0) & (s<=len),
     %disp( [ 'Pierces facet ' num2str(ifacet) ] )
     collision=1;
  end  % bottom of if s
  else
     collision=0;
  end  % bottom of if in

% bottom of for ifacet
end
% bottom of read_stl_binary
```

Appendix B

Inverse Kinematic Derivations

B.1 3 DOF Arm Kinematics

Below are all transformation matrices from the zero coordinate frame to the fourth coordinate frame

General form of translation along axis $\hat{Q}$ by distance r and rotation about same axis by $\hat{Q}$ [1]:

$$ {}^{0}_{1}T = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.1} $$

$$ {}^{1}_{2}T = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & a_1 \\ 0 & 0 & 1 & d_2 \\ -\sin\theta_2 & -\cos\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.2} $$

$$ {}^{2}_{3}T = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & a_2 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.3} $$

$$ {}^{3}_{4}T = \begin{bmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & a_3 \\ 0 & 0 & 1 & d_4 \\ -\sin\theta_4 & -\cos\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.4} $$

Finally, a point P is added to the end of the arm by attaching a coordinate frame at P that is a constant distance d_5 from the X_4 coordinate frame

$$ {}^{4}_{5}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.5} $$

Multiply all transformation matrices together to get a final expression from the 0 coordinate frame to the 5th coordinate frame.

$$ {}^{0}_{5}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T\,{}^{4}_{5}T \tag{B.6} $$

Substitute $r_i i$ and P_i for matrix entries:

$$ {}^{0}_{6}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & P_X \\ r_{21} & r_{22} & r_{23} & P_Y \\ r_{31} & r_{32} & r_{33} & P_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.7} $$

$$ \begin{aligned} P_X &= c\theta_1[a_1 + a_2 c\theta_2 + a_3\cos\theta_{23} - s\theta_{23}d_5 - d_4 s\theta_{23}] - d_2 s\theta_1 \\ P_Y &= s\theta_1[a_1 + a_2 c\theta_2 + a_3\cos\theta_{23} - d_4 s\theta_{23} - s\theta_{23}d_5] + c\theta_1 d_2 \\ P_Z &= d_1 - c\theta_{23}d_5 - d_4 c\theta_{23} - a_3 s\theta_{23} - a_2 s\theta_2 \end{aligned} \tag{B.8} $$

B.2 Kinetic Energy of RE2 Robotic Arm

This section describes the equations used to solve for the mass center velocities of each link. These velocities are then used to find the kinetic energy of a three-degree-of-freedom arm.

Using the transformation matrices derived in the section B.1, the global transformation matrices for local coordinate frames 1, 2 and 3 are derived.

$$ {}^{0}_{1}T = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.9} $$

$$ {}^{0}_{2}T = \begin{bmatrix} \cos\theta_2\cos\theta_1 & -\cos\theta_1\sin\theta_2 & -\sin\theta_1 & a_1\cos\theta_1 - d_2\sin\theta_1 \\ cos\theta_2\sin\theta_1 & -sin\theta_1\sin\theta_2 & \cos\theta_1 & d_2\cos\theta_1 + a_1\sin\theta_1 \\ -\sin\theta_2 & -\cos\theta_2 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.10} $$

$$ {}^{0}_{3}T = {}^{0}_{2}T\,{}^{2}_{3}T \tag{B.11} $$

Using the transformation matrices, the global positions of each mass center can then be calculated.

$$ {}^{0}r_1 = {}^{0}_{1}T^{1}r_1 = {}^{0}_{1}T \begin{bmatrix} 0 \\ c_1 \\ 0 \\ 1 \end{bmatrix} \tag{B.12} $$

$$ {}^{0}r_2 = {}^{0}_{2}T^{2}r_2 = {}^{0}_{2}T \begin{bmatrix} c_2 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{B.13} $$

$$ {}^{0}r_3 = {}^{0}_{3}T^{3}r_3 = {}^{0}_{3}T \begin{bmatrix} 0 \\ c_3 \\ 0 \\ 1 \end{bmatrix} \tag{B.14} $$

$$ {}^{0}d_1 = {}^{0}_{1}T^{1}d_1 = {}^{0}_{1}T \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{B.15} $$

$$ {}^0d_2 = {}^0_2T^2 d_2 = {}^0_1T \begin{bmatrix} a_2 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{B.16} $$

$$ {}^0d_2 = {}^0_2T^2 d_2 = {}^0_1T \begin{bmatrix} a_2 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{B.17} $$

$$ {}^0d_3 = {}^0_3T^3 d_3 = {}^0_3T \begin{bmatrix} 0 \\ c_4 \\ 0 \\ 1 \end{bmatrix} \tag{B.18} $$

Angular linkage velocities are:

$$ \begin{aligned} {}^0w_1 &= \dot{\theta}_1 \hat{k}_0 \\ {}^1w_2 &= \dot{\theta}_2 \hat{k}_1 \\ {}^2w_3 &= \dot{\theta}_2 \hat{k}_2 \end{aligned} \tag{B.19} $$

$$ \begin{aligned} {}^0\tilde{w}_2 &= {}^0\tilde{w}_1 + {}^0_1R^1 \tilde{w}_2 {}^0_1R^T \\ {}^0\tilde{w}_3 &= {}^0\tilde{w}_2 + {}^0_2R^2 \tilde{w}_2 {}^0_2R^T \end{aligned} \tag{B.20} $$

Where ${}^0\tilde{w}_1$ is the skew symmetric matrix of the velocity vector, and ${}^0_iR^T$ is the rotation matrix from the origin to frame i. The mass moment matrix in the global coordinate frame:

$$ {}^0_iI = {}^0_1R \begin{bmatrix} I_{X,i} & 0 & 0 \\ 0 & I_{Y,i} & 0 \\ 0 & 0 & I_{Z,i} \end{bmatrix} {}^0_1R^T \tag{B.21} $$

The global velocity of each mass center can be described by:

$$ \begin{aligned} {}^0v_i &= \tfrac{{}^0d}{dt} {}^0r_i \\ {}^0\dot{d}_i &= \tfrac{{}^0d}{dt} {}^0d_i \end{aligned} \tag{B.22} $$

The final expression for the kinetic energy of the entire RE2 arm is:

$$\begin{aligned} K_{Arm} &= \tfrac{1}{2}{}^0v_1^T m_1{}^0v_1 + \tfrac{1}{2}{}^0v_2^T m_2{}^0v_2 + \tfrac{1}{2}{}^0v_3^T m_3{}^0v_3 ... \\ &... + \tfrac{1}{2}{}^0\dot{d}_3^T m_0{}^0\dot{d}_3 + \tfrac{1}{2}{}_0w_1^{To} I_{10} w_1 + \tfrac{1}{2}{}_0w_2^{To} I_{20} w_2 ... \\ &... + \tfrac{1}{2}{}_0w_3^{To} I_{30} w_3 \end{aligned} \tag{B.23}$$

Bibliography

[1] Craig, J. (2005) *Introduction to Robotics Mechanics and Control*, 3rd ed., Pearson Prentice Hall, New Jersey.

[2] Sommer, H. (2014), "Geometry of Points, Rays, Planes and Cylinders," Geometry Notes.

[3] openJAUS LLC (2007) *The Joint Architecture for Unmanned Systems: Reference Architecture Specification*, *Tech. rep.*

[4] Stulp, F. and F. Fedrizzi (2009) "Learning and Performing Place-Based Mobile Manipulation," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

[5] Stulp, F. (2009) "Action Related Place-Based Mobile Manipulation," in *22009 IEEE 8th Annual Conference on Development and Learning*, pp. 3115–3120.

[6] Berenson, D. and J. Kuffner (2008) "An Optimization Approach to Planning for Mobile Manipulation," in *2008 IEEE International Conference on Robotics and Automation*, pp. 1187–1192.

[7] Zacharias, F. and M. Beetz (2008) "Positioning Mobile Manipulators to Perform Constrained Linear Trajectories," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2578–2584.

[8] Gaschler, A. and R. Petrick (2013) "KVP: A Knowledge of Volumes Approach to Robot Task Planning," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 202–208.

[9] Berenson, D. and S. Siddjartha (2009) "Manipulation Planning with Workspace Goal Regions," in *2009 IEEE International Conference on Robotics and Automation*, pp. 618–624.

[10] Bley, F. and V. Schmitgel (2006) "Mobile Manipulation Based on Generic Object Knowledge," in *15th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 411–416.

[11] Zacharias, F. and M. Beetz (1991) "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation," in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pp. 1398–1404.

[12] Ge, S. S. and Y. J. Cui (2000) "New Potential Functions for Mobile Robot Path Planning," *IEEE Transactions on Robotics and Automation*, **16**(5), pp. 615–620.

[13] Zhang, X. and Y. Fang (2013) "Set-Oriented Optimal Path Planning of Mobile Robots by a Polynomial Rootfinder," in *2013 IEEE International Conference on Control Applications (CCA)*, pp. 778–783.

[14] Colbaugh, R. (1998) "Adaptive Stabilization of Mobile Manipulators," *Journal of Robotic Systems*, **15**, pp. 511–523.

[15] Seraji, H. (1993) "Motion Control of Mobile Manipulation," *Proceedings of 1993 IEEE/RSJ Conference on Intelligent Vehicle Robotics*, pp. 2056–2063.

[16] Ben-Tzvi, P. (2010) "Experimental Validation and Field Performance Metrics of a Hybrid Robot Mechanism," *Journal of Field Robotics*, **27**, pp. 250–266.

[17] Gardner, J. and S. Velinsky (2010) "Kinematics of Mobile Manipulators and Implications for Design," *Journal of Robotic Systems*, **27**, pp. 309–312.

[18] Jazar, R. (2010) *Theory of Applied Robotics*, 2nd ed., Springer, New York.

[19] Brennan, S. (2014), "Methods for Robot Path Planning," ME 456 Lecture.

[20] Open-Source, "gazebosim.org Website," .
URL `http://gazebosim.org/`

[21] OpenSource, "ros.org Website," .
URL `http://wiki.ros.org/`

[22] Benhabib, B., A. A. Goldenberg, and R. G. Fenton (1985) "A solution to the inverse kinematics of redundant manipulators," *Journal of Robotic Systems*, **2**(4), pp. 373–385.
URL `http://dx.doi.org/10.1002/rob.4620020404`

[23] *Interface Control Document for Automatic Arm, Tech. rep.*, RE2 Inc.