LOW-COST TERRESTRIAL DEMONSTRATION OF AUTONOMOUS SATELLITE PROXIMITY OPERATIONS

THESIS

Zackary Hewitt, First Lieutenant, USAF

AFIT-ENG-MS-21-M-047

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this document are those of the author and do not reflect the official policy or position of the United States Air Force, the United States Department of Defense or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT-ENG-MS-21-M-047

LOW-COST TERRESTRIAL DEMONSTRATION OF AUTONOMOUS SATELLITE PROXIMITY OPERATIONS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Zackary Hewitt, B.S.E.E.

First Lieutenant, USAF

March 25, 2021

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT-ENG-MS-21-M-047

LOW-COST TERRESTRIAL DEMONSTRATION OF AUTONOMOUS SATELLITE PROXIMITY OPERATIONS

THESIS

Zackary Hewitt, B.S.E.E.
First Lieutenant, USAF

Committee Membership:

Robert Leishman, Ph.D
Chair

Maj Costantinos Zagaris, Ph.D
Member

Maj Joshuah Hess, Ph.D
Member

AFIT-ENG-MS-21-M-047

Abstract

The lack of satellite servicing capabilities significantly impacts the development and operation of current orbital assets. With autonomous solutions under consideration for servicing, the purpose of this research is to build and validate a low-cost hardware platform to expedite the development of autonomous satellite proximity operations. This research aims to bridge the gap between simulation and existing higher fidelity hardware testing with an affordable alternative. An omnidirectional variant of the commercially available TurtleBot3 mobile robot is presented as a 3-DOF testbed that demonstrates a satellite servicing inspection scenario. Reference trajectories for the scenario are generated via optimal control using the commercial solver GPOPS-II, and results from simulation and hardware demonstration are presented. Recommendations are then given for using the platform as a rapid method for experimentally verifying various satellite control algorithms.

Table of Contents

Page

Abstract iv

List of Figures vii

List of Tables xii

I. Introduction 1

1.1 Motivation 1
1.2 Problem Statement 2
1.3 Research Objectives 2
1.4 Assumptions and Limitations 3
1.5 Methodology Overview 4
1.6 Thesis Contributions 5
1.7 Document Overview 5

II. Background and Literature Review 7

2.1 Satellite Dynamics 7
2.1.1 Orbital Environment 7
2.1.2 Relative Orbital Motion 9
2.1.3 3-DOF Formulation 11
2.2 Mobile Robot Kinematics 14
2.2.1 Global and Local Reference Frames 14
2.2.2 Wheel Constraints 16
2.2.3 Robot Maneuverability 19
2.2.4 Equations of Motion 20
2.3 Optimal Control Methods 23
2.3.1 Objective Functions 24
2.3.2 Constraints 24
2.3.3 GPOPS-II 25
2.4 The Robot Operating System 26
2.4.1 ROS Nodes and Topics 26
2.4.2 ROS Packages and Services 27
2.5 Chapter Summary 28

III. Methodology 30

3.1 Hardware Systems 30
3.1.1 Mobile Robot 30
3.1.2 Motion Capture System 37
3.2 Testbed Network 42
3.2.1 Network Overview 42

Page

3.2.2 Computing Environment . . . 42
3.3 Satellite Control Simulation . . . 46
3.3.1 Example Satellite Servicing Scenario . . . 46
3.3.2 Time-Optimal Reference Trajectories . . . 48
3.3.3 Control Algorithm . . . 49
3.4 Chapter Summary . . . 53

IV. Results and Analysis . . . 54

4.1 Scenario Map Overview . . . 54
4.2 Test Cases . . . 56
4.2.1 Test Case 1: Stationary Chief . . . 57
4.2.2 Test Case 2: Rotating Chief 1 . . . 65
4.2.3 Test Case 3: Rotating Chief 2 . . . 71
4.2.4 Test Case 4: Approximated Dynamics 1 . . . 77
4.2.5 Test Case 5: Approximated Dynamics 2 . . . 81
4.2.6 Test Case 6: Double Mass Deputy . . . 85
4.2.7 Test Case 7: Half Mass Deputy . . . 89
4.3 Full Scale Testing . . . 93
4.4 Error Analysis . . . 101
4.4.1 Open Loop vs Closed Loop . . . 101
4.4.2 Simulation vs Hardware . . . 102
4.4.3 Full Scale vs Scaled Down . . . 102
4.5 Chapter Summary . . . 104

V. Conclusions . . . 105

5.1 Thesis Contributions . . . 106
5.2 Future Work . . . 107

Appendix A. Test Case 1 . . . 108

Appendix B. Test Case 2 . . . 114

Appendix C. Test Case 3 . . . 120

Appendix D. Test Case 4 . . . 126

Appendix E. Test Case 5 . . . 132

Appendix F. Test Case 6 . . . 138

Appendix G. Test Case 7 . . . 144

Bibliography . . . 150
Acronyms . . . 153

List of Figures

Figure		Page
1	Earth-Centered-Inertial Reference Frame	8
2	Local Vertical, Local Horizontal Reference Frame	10
3	Mobile Robot aligned with a global axis	15
4	Fixed Standard Wheel Parameters	17
5	Swedish Wheel Parameters	18
6	Constrained Maneuverability	20
7	Omnidirectional Motion	21
8	Three-wheeled omnidirectional robot geometry	22
9	ROS rqt_graph	27
10	ROS Turtlesim Example	28
11	Turtlebot versions	31
12	60 mm Aluminum Swedish Wheel	32
13	Waffle-plate Layer Assembly	33
14	Turtlebot3 Omni Layer	34
15	Omnibot Servo Mounting Brackets	35
16	Swedish Wheel Adapter	35
17	Assembled Omnibot	36
18	Raspberry Pi and OpenCR boards	37
19	Gazebo Simulator	38
20	ANT VICON Chamber	39
21	VICON Tracker Camera Arrangement	40
22	VICON Tracker Omnibot Object	40

Figure Page

23 VICON Example Pose Data 41

24 Master Laptop Active Terminals 43

25 Scenario Map Initial 55

26 Scenario Map in Progress 56

27 Test 1 Optimal Solution: Trajectory 58

28 Test 1 Optimal Solution: Control 59

29 Test 1 Optimal Solution: State Velocities 60

30 Test 1 Time Optimal Solution 61

31 Test 1 Simulated Open Loop Control 62

32 Test 1 Hardware Video Frames 63

33 Test 1 Hardware Open Loop Control 64

34 Test 2 Optimal Solution: Trajectory 66

35 Test 2 Time Optimal Solution 67

36 Test 2 Hardware Open Loop Control 68

37 Test 2 Hardware Closed Loop Control 70

38 Test 2 Hardware Feedback Signal 71

39 Test 3 Optimal Solution: Trajectory 72

40 Test 3 Time Optimal Solution 73

41 Test 3 Hardware Closed Loop Control 74

42 Test 3 Simulated Closed Loop Control 75

43 Compared Feedback Signals in Test 3 76

44 Test 4 Time Optimal Solution 78

45 Compared Dynamics Test 2 vs 4 79

46 Test 4 Hardware Closed Loop Control 80

Figure Page

47 Test 5 Time Optimal Solution 82

48 Compared Dynamics Test 3 vs 5 83

49 Test 5 Hardware Closed Loop Control 84

50 Test 6 Time Optimal Solution 86

51 Compared Mass Test 3 vs 6 87

52 Test 6 Hardware Closed Loop Control 88

53 Test 7 Time Optimal Solution 90

54 Compared Mass Test 3 vs 7 91

55 Test 7 Hardware Closed Loop Control 92

56 Test 1 MAV Lab 80% Scaling 94

57 Test 2 MAV Lab 80% Scaling 95

58 Test 3 MAV Lab 80% Scaling 96

59 Test 1 Under Hardware Velocity Limit 97

60 Test 2 Over Hardware Velocity Limit 98

61 Test 3 Over Hardware Velocity Limit 98

62 Test 2 MAV Lab 70% Scaling 99

63 Test 3 MAV Lab 60% Scaling 100

A1 Test 1 Time Optimal Solution 109

A2 Test 1 Simulated Open Loop Control 110

A3 Test 1 Hardware Open Loop Control 111

A4 Test 1 Simulated Closed Loop Control 112

A5 Test 1 Hardware Closed Loop Control 113

B1 Test 2 Time Optimal Solution 115

B2 Test 2 Simulated Open Loop Control 116

Figure Page

B3 Test 2 Hardware Open Loop Control 117

B4 Test 2 Simulated Closed Loop Control 118

B5 Test 2 Hardware Closed Loop Control 119

C1 Test 3 Time Optimal Solution 121

C2 Test 3 Simulated Open Loop Control 122

C3 Test 3 Hardware Open Loop Control 123

C4 Test 3 Simulated Closed Loop Control 124

C5 Test 3 Hardware Closed Loop Control 125

D1 Test 4 Time Optimal Solution 127

D2 Test 4 Simulated Open Loop Control 128

D3 Test 4 Hardware Open Loop Control 129

D4 Test 4 Simulated Closed Loop Control 130

D5 Test 4 Hardware Closed Loop Control 131

E1 Test 5 Time Optimal Solution 133

E2 Test 5 Simulated Open Loop Control 134

E3 Test 5 Hardware Open Loop Control 135

E4 Test 5 Simulated Closed Loop Control 136

E5 Test 5 Hardware Closed Loop Control 137

F1 Test 6 Time Optimal Solution 139

F2 Test 6 Simulated Open Loop Control 140

F3 Test 6 Hardware Open Loop Control 141

F4 Test 6 Simulated Closed Loop Control 142

F5 Test 6 Hardware Closed Loop Control 143

G1 Test 7 Time Optimal Solution 145

Figure Page

G2 Test 7 Simulated Open Loop Control 146

G3 Test 7 Hardware Open Loop Control 147

G4 Test 7 Simulated Closed Loop Control 148

G5 Test 7 Hardware Closed Loop Control 149

List of Tables

Table		Page
1	Common ROS Commands	26
2	Turtlebot3 Waffle Kit Core Components	31
3	Static IP Address Assignment	42
4	MATLAB Controller ROS Topics	45
5	Scenario Constants Base Configuration	49
6	Seven Test Cases	57
7	RMSE Open Loop vs Closed Loop	102
8	RMSE Simulation vs Hardware	103
9	RMSE Full Scale vs Scaled Down	103

LOW-COST TERRESTRIAL DEMONSTRATION OF AUTONOMOUS SATELLITE PROXIMITY OPERATIONS

I. Introduction

1.1 Motivation

As commercial, scientific, and military interest in space expands, the ability to revisit assets on orbit for servicing has become a necessity [1]. However, the complexity of these servicing missions is exceeding the capabilities of humans to manually intervene and reveals the need for autonomous solutions.

The historic context for the space developments leading up to the first autonomous orbital rendezvous demonstrations is provided by Woffinden in "Navigating the Road to Autonomous Orbital Rendezvous" [2]. Woffinden recounts the United States' and Russia's rendezvous missions starting from the 1960s, including examples of on-orbit servicing in the early Gemini missions, the shuttle missions to repair the Hubble Telescope, and on-going missions on the International Space Station. While these missions have demonstrated successful on-orbit servicing, all of these missions had humans in the loop. As repair mission requirements increase in complexity, the ability to use humans to repair on-orbit assets is diminishing. Further, these missions have all occurred close to the earth in Low Earth Orbit (LEO). In the case of large, expensive, and strategically important satellites in Geosynchronous Equatorial Orbit (GEO), it may not be feasible to send human astronauts for GEO repair missions [1]. For teleoperation from a ground station, as distances from Earth increase the latency involved may be too high to handle the complex maneuvers of a repair mission.

Therefore, the need for an on-board autonomous solution is clear.

Autonomous Guidance, Navigation, and Control (GNC) algorithms can be simulated to demonstrate sufficient performance before being deployed to an operational environment. However, including hardware testing is desirable to demonstrate that the algorithm can also perform in real-time on actual hardware. The purpose of this work is to build a framework that can abstract the orbital domain from an autonomous on-orbit servicing satellite controller to enable testing on a ground-based platform. This framework yields a flexible and cost-effective method for conducting rapid prototyping of novel autonomous satellite behaviors prior to funding higher fidelity testing and ultimately orbital demonstration.

1.2 Problem Statement

One source of difficulty in developing autonomous algorithms for space applications is performing hardware testing to experimentally validate simulated results. With the high cost of launching developmental assets to orbit, engineers must rely on terrestrial methods for a majority of initial hardware testing. Terrestrial methods often involve the use of highly specialized laboratory space and state-of-the-art equipment; therefore, only a handful of labs have the resources to perform this level of experimentation [3, 4, 5]. This research does not aim to replace higher fidelity platforms fully. This research strives to accelerate autonomous control development by bridging the gap between simulation and higher fidelity hardware testing with an alternative low-cost setup using open-source wheeled mobile robots.

1.3 Research Objectives

The primary objective of this research is to build and validate a hardware platform for rapidly testing various control methods in the development of autonomous satellite

proximity operations. The first objective is to generate a set of reference trajectories of a deputy satellite performing a time optimal inspection scenario about a rotating chief satellite. The second objective is to design and assemble a terrestrial hardware platform that is capable of demonstrating the reference scenario. The third objective is to perform the inspection scenario on the hardware and compare its performance against the time optimal reference trajectories. The result is a local platform that can be used to physically demonstrate current and future GNC research that would otherwise be restricted to simulation.

1.4 Assumptions and Limitations

The Aerospace Corporation divided on-orbit servicing capabilities into seven categories: non-contact support, orbit modification, refueling, upgrade, repair, assembly, and debris mitigation [6]. Non-contact support includes inspection of a client's satellite and serves as the example mission scenario for this research. The other servicing missions involve the servicer mating with the client satellite and introduces contact dynamics that is not investigated in this work.

Satellite proximity operations in the literature are often further separated into two categories: cooperative and noncooperative [7]. Opromolla's research defines cooperative satellites as being able to provide the chaser satellite with information to estimate the pose (relative position and attitude) of the target satellite. Inspecting a cooperative satellite allows this research to focus on autonomous control without issues related to state estimation.

Further, the research is performed on a mobile robot system with three degrees of freedom (DOF) via two translational DOF and one rotational DOF. This is contrasted to a real satellite in orbit with six DOF, via three translational DOF and three rotational DOF. This connection is possible due to the Clohessy-Wiltshire (CW) equations

of relative orbital motion that show that the radial (x) and in-track (y) components of relative orbital motion can be decoupled from the cross-track (z) component in the Local Vertical, Local Horizontal (LVLH) reference frame [8]. However, the CW equations impose a few assumptions, including that the chief satellite's orbit is circular and that the distance between the chief and deputy remains sufficiently small compared to the distance between the chief and the Earth. These assumptions fit well with a typical GEO satellite orbital profile, and, therefore, this research investigates satellite inspection maneuvers within 10 km of a chief satellite stationed in GEO.

1.5 Methodology Overview

First, time optimal reference trajectories for a satellite inspection scenario are generated via a commercial optimal control solver known as GPOPS-II [9]. In this research, GPOPS outputs full state information about a servicer satellite with one rotational and two translational DOF. The optimal solution adheres to a collision avoidance constraint, thrust and torque control constraints, and terminal constraints such that the servicer ends at relative rest within a cone originating from the client at a desired distance and with the servicer pointing towards the client.

Second, the hardware platform is constructed as a holonomic mobile robot based on the TurtleBot open-source robotic platform. This variant of the TurtleBot uses three omni-wheels separated by 120 degrees to produce holonomic motion that follows the motion of a double integrator or a free-flying robot [10]. The omni-wheel TurtleBot variant uses the Robot Operating System (ROS) to connect the on-board Raspberry Pi to a Linux laptop for telemetry, a windows desktop hosting a motion capture system, and to a MATLAB or Simulink control algorithm for testing.

Third, the reference scenarios are performed on the hardware in the Autonomy and Navigation Technology (ANT) center VICON motion capture chamber that provides

pose measurements to compare the performance of the testbed with the generated reference trajectories.

1.6 Thesis Contributions

- Built a low-cost three degrees of freedom (3-DOF) hardware testbed for autonomous satellite servicing research based on a variant of the commercially available TurtleBot mobile robot. The testbed provides a lower-fidelity alternative to limited state-of-the-art facilities for rapid control algorithm prototyping.

- Set up a ROS network that enables communication between the mobile robot, the satellite control simulation that outputs the guidance trajectories and control commands, and the motion capture system that relays navigation measurements for feedback control and testbed performance analysis.

- Provided an example satellite servicing scenario involving the inspection of a rotating chief satellite that demonstrates the capabilities of the testbed. The example scenario and example feedback controller are designed to be modified or replaced to aid the development of future controls research.

- Performed simulation and hardware testing to validate the performance of the testbed across seven test cases based on the example scenario. Testing is performed in two different motion capture facilities to highlight the flexibility of the testbed in different environments and to show the effect of distance scaling in smaller laboratories.

1.7 Document Overview

This thesis is further organized into five chapters as follows:

Chapter II provides background information on satellite dynamics, including the orbital environment, relative orbital dynamics, and a 3-DOF problem formulation. The chapter further details relevant mobile robot dynamics for comparison. Then, optimal control methods are presented. Lastly, ROS is briefly introduced.

Chapter III describes the developed testbed framework, including the control simulation, the ROS network setup, the mobile robot hardware implementation, and the motion capture system.

Chapter IV details the results obtained from performing a satellite servicing inspection scenario on the hardware testbed. The results are compared to time optimal control reference trajectories for the servicing scenario.

Chapter V summarizes the findings of the research and emphasizes the ability of low-cost hardware testing to enable rapid autonomous algorithm development. The chapter concludes with recommendations on how to improve the testbed and suggests additional scenarios for demonstration.

II. Background and Literature Review

This chapter provides the background necessary to abstract the motion of a satellite to a wheeled mobile robotic testbed for hardware demonstration. First, the models for relative orbital dynamics of a satellite and holonomic motion of a three omni-directional wheeled mobile robot are introduced. Next, optimal control methods and a commercial solver are discussed. Finally, the framework that connects the spacecraft simulation and the robotic hardware are introduced.

2.1 Satellite Dynamics

This section presents the background on satellite orbital dynamics that is relevant to the development of an example satellite control system. A description of the orbital environment, a classic set of relative satellite motion equations, and a three degrees of freedom (3-DOF) formulation is presented to scope the breadth of satellite dynamics to this research.

2.1.1 Orbital Environment

There is a wide variety in applications of space assets, and a core part of understanding the behavior of a satellite is in understanding its orbital regime. The following section presents a common reference frame and the classical orbit elements used to describe the motion of a satellite in orbit around the Earth.

2.1.1.1 Earth-Centered-Inertial Reference Frame

A common reference frame for the description of satellite motion is the Earth-Centered-Inertial (ECI) reference frame. In the ECI frame, the $\hat{\mathbf{x}}$ vector points from the Earth's center of mass to the vernal equinox, the $\hat{\mathbf{z}}$ vector is normal to the equa-

torial plane through the north pole, and the $\hat{\mathbf{y}}$ vector is perpendicular to the others along the equatorial plane [11].

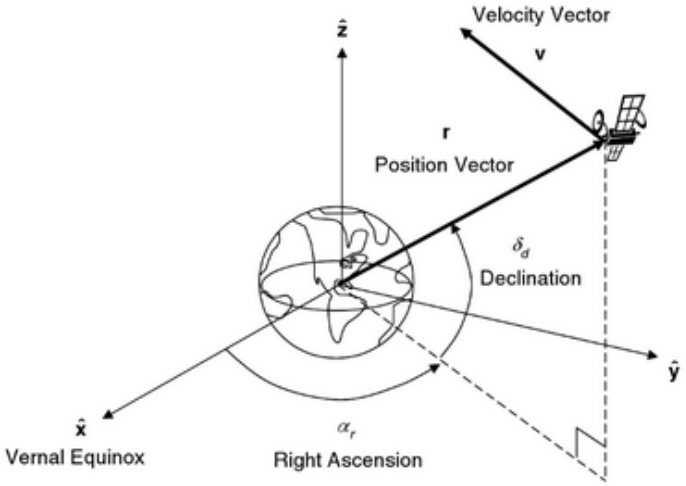


Figure 1: Earth-Centered-Inertial Reference Frame [11]

2.1.1.2 Classical Orbital Elements

Within the ECI reference frame, the orbit of a given satellite can be defined by the six Classical Orbital Elements (COEs): semi-major axis (a), eccentricity (e), inclination (i), right ascension of the ascending node (Ω), argument of perigee (ω), and true anomaly (ν).

The satellite in the inspection scenario of this research is assumed to be in a Geosynchronous Equatorial Orbit (GEO) with a period of one sidereal day, a semi-major axis of 42,164 km (approximately 35,786 km above sea level), an eccentricity of 0 (circular), and an inclination of 0 deg (on the equatorial plane). Both the right ascension of the ascending node and the argument of perigee are undefined in this GEO case due to both 0 eccentricity and inclination. Lastly, the true anomaly varies with time over a full revolution of the orbit. This specific orbital profile was chosen for this research as it is a strategically important orbit for many critical space

assets that could benefit from satellite servicing [1]. A GEO mission also makes teleoperation from the ground more difficult due to latency and involves a worksite that is inaccessible to humans [1]. Additionally, the characteristics of GEO enable use of the Clohessy-Wiltshire simplified model of relative orbital motion, which is further detailed in Section 2.1.2.2 below.

2.1.2 Relative Orbital Motion

The previous section presents a method for describing an individual satellite with respect to a celestial body. However, when studying the motion of two or more satellites in close proximity orbiting the same celestial body it can be much simpler to analyze their motion relative to each other. The following sections present an alternative reference frame and relevant assumptions for the use of a classical set of linear equations of relative orbital motion.

2.1.2.1 Local Vertical, Local Horizontal Frame

Instead of using the ECI frame to independently analyze the orbital motion of both a client and a servicer as individual satellites, the Local Vertical, Local Horizontal (LVLH) frame is often used to describe the relative motion of two neighboring satellites [11]. Within the literature on relative orbital motion, the client satellite is also referred to as the chief and the servicer satellite is referred to as the deputy [6]. The origin of the LVLH frame is at the center of mass of the chief, the x-axis points outward along the orbit's radius, the z-axis is perpendicular to the chief's orbital plane in the direction of the orbital angular momentum vector, and the y-axis completes the right-handed coordinate system [11]. For a circular orbit, the y-axis points in the direction of the chief's velocity vector.

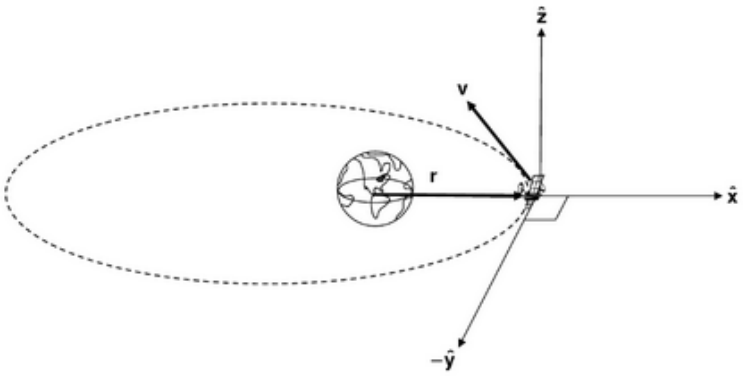


Figure 2: Local Vertical, Local Horizontal Reference Frame [11]

2.1.2.2 Clohessy-Wiltshire Equations of Motion

The Clohessy-Wiltshire (CW) equations of relative orbital motion were developed in the 1960s to analyze spacecraft rendezvous scenarios [8]. The CW equations are written in the LVLH frame and are linearized about the origin with the assumption that the chief's orbit is circular and that the deputy's orbit is only slightly elliptic. To maintain fidelity, the distance between the chief and the deputy must remain sufficiently small compared to the distance between the chief and the Earth. The satellite servicing scenarios investigated in this research remain within 10 km, and a semi-major axis in excess of 40,000 km at GEO satisfies this requirement. The CW equations are given below

$$\ddot{x} = 3n^2x + 2n\dot{y}, \tag{1}$$

$$\ddot{y} = -2n\dot{x}, \tag{2}$$

$$\ddot{z} = -n^2z, \tag{3}$$

where x is the radial component, y is the in-track component, and z is the cross-track component. The mean motion of the chief is defined as $n = \sqrt{\mu/a^3}$ where μ is the standard gravitational parameter and a is the semi-major axis of the orbit. The

formulation of the CW equations above reveals that the radial and in-track (x and y axes) components are coupled, but the cross-track (z-axis) component is decoupled and can be solved independently. This enables the study of the translational behavior of relative satellite motion where the satellites are essentially restricted to the orbital plane with only two degrees of freedom (2-DOF).

2.1.2.3 Free-Flying Model

A further simplified model of relative orbital motion can be made by discarding negligible terms in the CW equations. At GEO, the mean motion is small relative to the acceleration terms and can be ignored over sufficiently short time periods [10]. When the mean motion terms are removed from the equations of motion, the model essentially reduces to a double integrator as shown below

$$\ddot{x} = a_x = \frac{F_x}{m}, \tag{4}$$

$$\ddot{y} = a_y = \frac{F_y}{m}, \tag{5}$$

$$\ddot{z} = a_z = \frac{F_z}{m}, \tag{6}$$

where each acceleration term can be simply described by the control force acting along each axis and the mass of the servicer given by m. This double integrator (DI) style model is also known as a free-flying model [12]. This linearized model can then be used as an approximation for rapid trajectory generation for close proximity servicing maneuvers that occur over the course of minutes.

2.1.3 3-DOF Formulation

A few restrictions must be made to accurately demonstrate the behavior of a satellite on a ground-based platform with a reduced number of degrees of freedom (DOF).

A satellite in orbit has six DOF (three translational DOF and three rotational DOF), but the wheeled robot used in this research has only three DOF (two translational DOF and one rotational DOF). The following sections show how the satellite motion can be reduced to a 3-DOF problem.

2.1.3.1 Planar Movement

The CW equations above in Section 2.1.2.2 show how the radial and in-track (x and y) components are decoupled from the cross-track (z) component of acceleration. Therefore, control can be freely applied in the radial and in-track directions without affecting the cross-track position. This is convenient as the satellite motion can be analyzed with only two translational DOF while no force is applied in the z component:

$$\ddot{x} = \frac{F_x}{m}, \tag{7}$$

$$\ddot{y} = \frac{F_y}{m}, \tag{8}$$

$$\ddot{z} = 0. \tag{9}$$

2.1.3.2 Angular Movement

Although a satellite has three rotational DOF (roll, pitch, and yaw), the wheeled robot can only rotate along the axis perpendicular to the ground for a total of one rotational DOF. Therefore, the satellite is only controlled about the z-axis for rotation that is perpendicular to the two axes of planar movement in the two translational DOF formulation above in Section 2.1.3.1. The angular acceleration is given by

$$\ddot{\theta} = \frac{\tau}{I_z}, \tag{10}$$

where τ is the control torque, and I_z is the servicer's moment of inertia about the cross-track (z) axis. No control torque is applied along the radial or in-track axes, and it is assumed that the satellite maintains orientation about those axes.

2.1.3.3 State Dynamics

With the translational and rotational equations of motion defined above, the complete state dynamics for the satellite in 3-DOF are given as

$$\ddot{x} = 3n^2x + 2n\dot{y} + \frac{F_x}{m}, \tag{11}$$

$$\ddot{y} = -2n\dot{x} + \frac{F_y}{m}, \tag{12}$$

$$\ddot{\theta} = \frac{\tau}{I_z}, \tag{13}$$

where F_x and F_y are the components of the control force in the radial and in-track directions given by

$$F_x = F\cos(\theta), \tag{14}$$

$$F_y = F\sin(\theta). \tag{15}$$

In Equations (11) and (12), m is the mass of the servicer, τ is the control torque, and I_z is the servicer's moment of inertia about the cross-track (z) axis. The thruster is assumed to be aligned with the center of mass so that its force does not cause a torque. The mass and moment of inertia are assumed to remain constant, but the results of different servicers with different physical characteristics are presented in

Chapter IV. The linearized state-space representation is written as

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 \\ 0 & \frac{1}{m} & 0 \\ 0 & 0 & \frac{1}{I_z} \end{bmatrix} \mathbf{u}, \tag{16}$$

where the state and control vectors are defined as

$$\mathbf{x} = \left[x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}\right]^T, \tag{17}$$

$$\mathbf{u} = \left[F_x, F_y, \tau\right]^T. \tag{18}$$

2.2 Mobile Robot Kinematics

The previous section explained how satellite motion can be reduced to a 3-DOF problem for implementation on a ground-based platform. However, mobile-robots have additional unique features and constraints that must be reconciled. Additionally, a robot that is designed to represent the motion of a satellite must have an appropriate degree of maneuverability. The following subsections introduce relevant reference frames, concepts of robot maneuverability, and the equations of motion of a three-wheeled omnidirectional platform.

2.2.1 Global and Local Reference Frames

The definition of the mobile robot's equations of motion begins with the definition of an inertially fixed reference frame. The global reference frame is defined by forming an arbitrary inertial basis with the axes X_I and Y_I from some origin O [13]. The

position of the robot is defined by a reference point P on the robot, and the local reference frame is formed by a basis of X_R and Y_R from P in plane with the global frame, as shown in Figure 3 below.

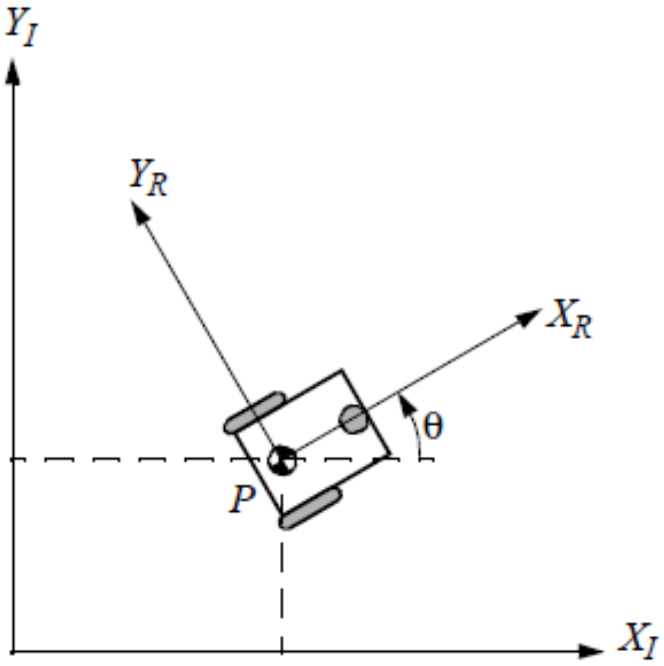


Figure 3: Mobile Robot aligned with a global axis [13]

Positions within the global reference plane relative to the origin are shown as ξ_I in Equation (19), and positions within the robot's local reference plane are shown as ξ_R in Equation (20) below:

$$\xi_I = \begin{bmatrix} x_I \\ y_I \end{bmatrix}, \tag{19}$$

$$\xi_R = \begin{bmatrix} x_R \\ y_R \end{bmatrix}. \tag{20}$$

With the angular difference between reference frames given by θ as shown in

Figure 3, positions can be mapped from the global to local reference frame by

$$\xi_R = R(\theta)\,\xi_I \tag{21}$$

using an orthonormal rotation matrix:

$$R(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}. \tag{22}$$

The kinematics of a mobile robot combine the motion of each wheel up to the motion of the full chassis in its local reference frame. The motion of the robot in its local reference frame can be mapped back to the global reference frame by

$$\dot{\xi}_I = R(\theta)^{-1}\,\dot{\xi}_R \tag{23}$$

using the inverse of the rotation matrix in Equation (24):

$$R(\theta)^{-1} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}. \tag{24}$$

Further, if the origin O is chosen as a reference point of another neighboring robot, the global reference frame resembles the LVLH frame used for relative orbital motion described above in Section 2.1.2.1.

2.2.2 Wheel Constraints

The kinematic model of a mobile robot chassis depends on the constraints imposed by the individual wheels [13]. The type of wheel used and the way it is installed in relation to the chassis imposes unique rolling and sliding constraints for each wheel.

The rolling constraint relates the wheel spin to the produced motion in the direction of the wheel plane and the sliding constraint constrains motion orthogonal to the wheel plane. In the case of the fixed standard wheel, the rolling constraint enforces the amount of wheel spin to be equal to the motion in the direction of the wheel plane, and the sliding constraint enforces zero motion orthogonal to the wheel plane. The fixed standard wheel rolling constraint is shown in Equation (25) and the fixed standard wheel sliding constraint is shown in Equation (26) below [13]:

$$[\sin(\alpha+\beta) - \cos(\alpha+\beta)(-l)\cos(\beta)]\,\dot{\xi}_R - r\dot{\phi} = 0, \tag{25}$$

$$[\cos(\alpha+\beta)\sin(\alpha+\beta)(l)\sin(\beta)]\,\dot{\xi}_R = 0, \tag{26}$$

where the position of the wheel is defined by the distance from the local reference frame origin, l, and the angle α, while the orientation of the wheel plane relative to the chassis is given by angle β as shown in Figure 4 [13]. Further, the change in wheel spin is defined by $\dot{\phi}$ while the wheel radius is defined by r.

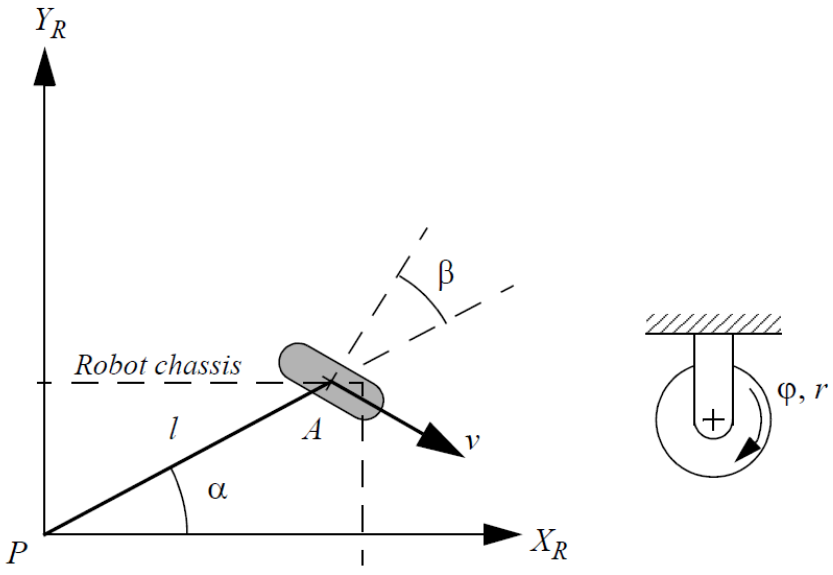


Figure 4: Fixed Standard Wheel Parameters [13]

Another type of wheel commonly used in mobile robotics is the Swedish wheel [13]. The Swedish wheel is similar to a fixed standard wheel but with rollers attached along the perimeter of the wheel at an angle of γ between the main wheel plane and axis of rotation of the rollers as shown in Figure 5. The Swedish wheel rolling constraint is shown in Equation (27), and the sliding constraint is shown in Equation (28) below [13]:

$$[\sin(\alpha+\beta+\gamma) - \cos(\alpha+\beta+\gamma)(-l)\cos(\beta+\gamma)]\,\dot{\xi}_R - r\dot{\phi}\cos(\gamma) = 0, \quad (27)$$

$$[\cos(\alpha+\beta+\gamma)\sin(\alpha+\beta+\gamma)(l)\sin(\beta+\gamma)]\,\dot{\xi}_R - r\dot{\phi}\sin(\gamma) - r_{sw}\dot{\phi}_{sw} = 0, \quad (28)$$

where r_{sw} is the radius of the roller and $\dot{\phi}_{sw}$ is the rate of the roller spin [13]. The roller is allowed to spin freely, so orthogonal motion is no longer constrained to zero, which enables the Swedish wheel to move omnidirectionally.

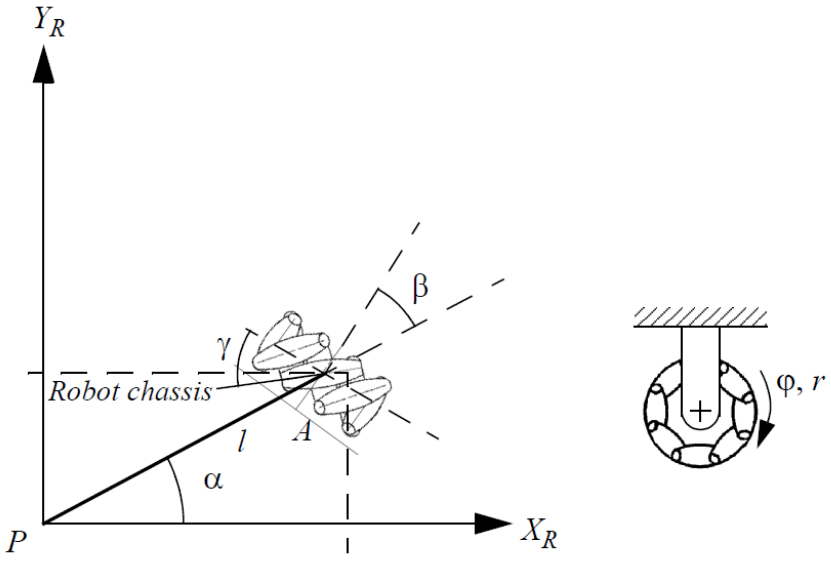


Figure 5: Swedish Wheel Parameters [13]

In this research, the 90 degree Swedish wheel variant is used where the rollers spin perpendicular to the main wheel plane so that the roller axis of rotation is parallel to

the wheel plane. In this case, $\gamma = 0$ degrees and the rolling and sliding constraints of Equations (27) and (28) reduce to [13]:

$$[\sin(\alpha+\beta) - \cos(\alpha+\beta)(-l)\cos(\beta)]\,\dot{\xi}_R - r\dot{\phi} = 0, \tag{29}$$

$$[\cos(\alpha+\beta)\sin(\alpha+\beta)(l)\sin(\beta)]\,\dot{\xi}_R - r_{sw}\dot{\phi}_{sw} = 0, \tag{30}$$

where the rolling constraint is the same as the fixed standard wheel in Equation (25), but the sliding constraint includes the free spinning roller. The addition of orthogonal motion due to the modified sliding constraint adds another degree of freedom to the Swedish wheel over the fixed standard wheel.

2.2.3 Robot Maneuverability

A robot's maneuverability is defined by its controllable degrees of freedom, which come from the degrees of mobility and degrees of steerability of its wheels [13]. As an example, Figure 6 shows a differential drive robot with two fixed standard wheels and an omnidirectional wheel. This mobile robot has two degrees of mobility from its wheels. While the robot only has two controllable degrees of freedom, it has three degrees of freedom within its workspace. The difference is subtle, but important. The robot can reach any point in x and y at an angle θ, but it cannot arrive at any point x and y with *any given* θ. As shown in Figure 6, due to the limited controllable degrees of freedom, the paths available to achieve a given pose are constrained.

An alternative robot configuration with three omnidirectional wheels is shown in Figure 7. This robot's wheels yield three degrees of mobility due to their orientation and ability to move orthogonally. This robot is able to directly control its velocity in x, y, and θ.

The number of independently achievable velocities is also known as a robot's differential degrees of freedom (DDOF) [13]. When a robot's DDOF equals its workspace

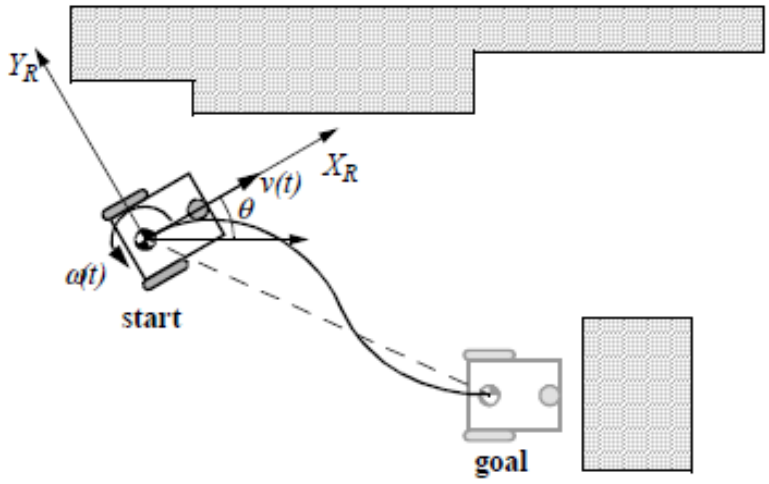


Figure 6: Constrained Maneuverability [13]

DOF it is known as a holonomic robot. In the case of a holonomic robot's workspace with $DOF = DDOF = 3$, the robot is known as an omnidirectional robot. An omnidirectional robot is able to maneuver to any position in its workspace without changing its orientation, which is useful in replicating the motion produced by the satellite dynamics in Section 2.1.

2.2.4 Equations of Motion

The platform selected for use in this research is a three-wheeled omnidirectional robot. Each wheel is separated by 120 degrees and is located a distance of L from the center. A geometric representation of the platform is shown below in Figure 8 [14].

The velocities of the mobile robot with respect to its local reference frame are given as V_x, V_y, and w. The individual wheel velocities are given as V_1, V_2, and V_3. Wheel velocities can be produced from local velocities by the following system of

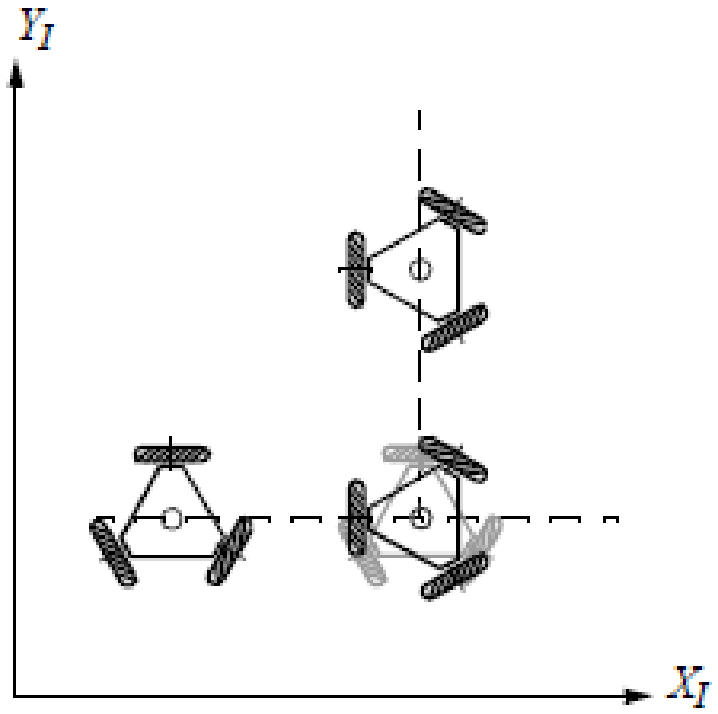


Figure 7: Omnidirectional Motion [13]

kinematic equations [15]:

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} -\sin(\theta) & \cos(\theta) & L \\ -\sin\left(\frac{\pi}{3}-\theta\right) & -\cos\left(\frac{\pi}{3}-\theta\right) & L \\ \sin\left(\frac{\pi}{3}+\theta\right) & -\cos\left(\frac{\pi}{3}+\theta\right) & L \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ w \end{bmatrix} . \tag{31}$$

If the orientation θ of the velocity vector v with respect to its local frame as shown in Figure 8 is fixed as $\theta = \frac{\pi}{6}$ so that V_3 remains parallel with V_x, Equation (31) reduces to

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} & \frac{\sqrt{3}}{2} & L \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & L \\ 1 & 0 & L \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ w \end{bmatrix} . \tag{32}$$

Equation (32) can then be used to find the necessary wheel velocities to produce

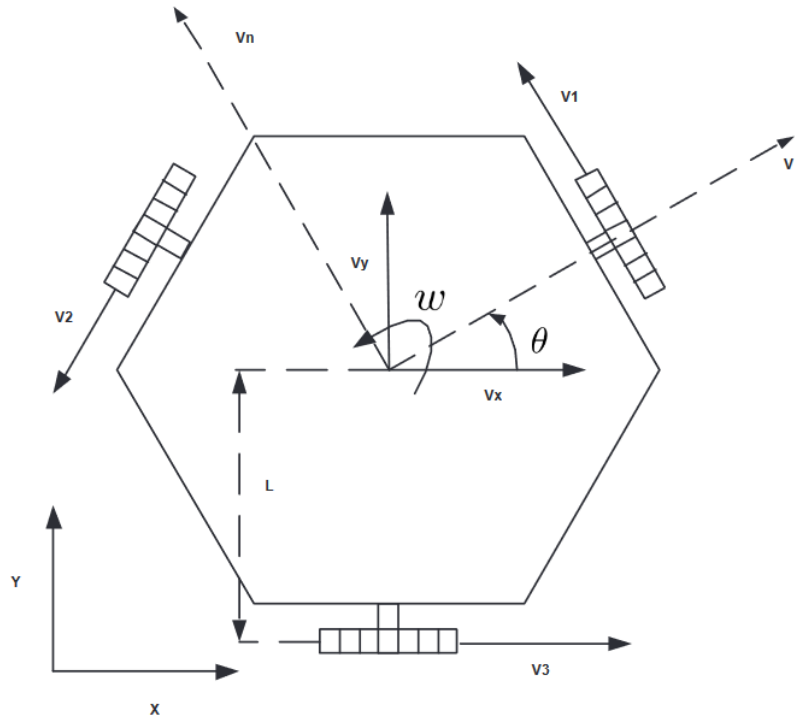


Figure 8: Three-wheeled omnidirectional robot geometry [14]

a desired velocity.

2.3 Optimal Control Methods

Optimal control involves finding a control law that minimizes a specified objective function while satisfying defined constraints [16]. Optimal control is notably useful in satellite applications due to its ability to find flight trajectories that minimize time or fuel used [17]. Constraints in the optimal control problem can also be used to enforce safety measures such as keep-out zones and control limits. A generic optimal control problem formulation is given as

$$\min_{t_f,u} g_b\left(x\left(t_f\right)\right) + \int_{t_0}^{t_f} g_i\left(x\left(t\right), u\left(t\right)\right) dt \tag{33}$$

subject to the following: for all $t \in [t_0, t_f]$,

$$\dot{x}\left(t\right) = \mathbf{f}\left(t, x\left(t\right), u\left(t\right)\right), \tag{34}$$

$$u\left(t\right) \in \mathbf{U}\left(t\right), \tag{35}$$

$$x(t) \in \mathbf{X}(t), \tag{36}$$

$$x(t_0) = x_0, \tag{37}$$

$$x(t_f) = x_f, \tag{38}$$

where the objective function in Equation (33) has a terminal cost component given by g_b and a running cost component in the integrand given by g_i [17]. The state of the system is given by x, the control input is u, and time is t. The state dynamics are defined by $\mathbf{f}$, and the sets of admissible control and state values are given by $\mathbf{U}$ and $\mathbf{X}$. Initial and final states are given by x_0 and x_f. The following sections present the components of an optimal control problem and also introduce a commercial optimization solver as a tool to numerically solve for optimal control solutions.

2.3.1 Objective Functions

The objective functional of an optimal control problem defines the quantity to be minimized. Two common types of optimal control problems are the minimum-time and the minimum-control-effort problems. In a minimum-time problem, the objective function is given simply as

$$J = t_f \tag{39}$$

where the extremal trajectory satisfies the terminal conditions in minimum time [16]. According to Pontryagin's Minimum Principle, this extremal trajectory uses maximum control authority [16]. In spacecraft applications it is also often important to conserve fuel due to finite on-board fuel limitations. To conserve fuel, the minimum-control-effort problem objective functional is given as

$$J = \int_{t_0}^{t_f} \|u(t)\|^2 \, dt. \tag{40}$$

2.3.2 Constraints

There are various constraints that can be used in the formulation of an optimal control problem to enable visual inspection of a point of interest on a client satellite while maintaining a safe distance. The first two presented below are control constraints where the servicer is assumed to have a maximum thrust force F_{max} and torque τ_{max} defined by Equations (41) and (42):

$$F \in [0, F_{max}], \tag{41}$$

$$|\tau| \leq \tau_{max}. \tag{42}$$

Next, a keep-out zone in the form of an ellipsoid can be used to enforce a safety constraint for collision avoidance [5, 18]. Since the problem has been reduced to

two translational DOF, a circle can be used to model the ellipsoid in the form of Equation (43) where R_{KOZ} is the radius of the keep-out circle, x and y are the position coordinates of the servicer at time t, and the circle is centered around the position of the client satellite at the origin:

$$R_{KOZ}^2 \leq x\left(t\right)^2 + y\left(t\right)^2 \tag{43}$$

for all $t \in [t_0, t_f]$.

Further, the problem can be constrained by the following terminal conditions to reach a desired end state:

$$R_G^2 \geq \left(x_f\right)^2 + \left(y_f\right)^2, \tag{44}$$

$$\left[\dot{x}_f, \dot{y}_f, \dot{\theta}_f\right] = \left[0, 0, 0\right]. \tag{45}$$

where Equation (44) has the servicer end within a desired distance from the client given by R_G and Equation (45) has the servicer end at relative rest.

2.3.3 GPOPS-II

Real-world problems often have nonlinearities that make solving an optimal control problem with indirect methods through the use of the calculus of variations or other classical methods intractable. Therefore, numerical methods are often used to solve this class of problem that may not have an analytical solution. GPOPS-II is a commercial solver that uses variable-order gaussian quadrature methods to numerically solve an optimal control problem that is approximated as a sparse nonlinear programming (NLP) problem [9]. The solver returns the optimized value of the objective function as well as the time, state, and control information at each collocation point of the solution. While the resulting solution is an open loop solution to the

optimal control problem, its implementation as a component of both an open loop controller and a closed loop controller is detailed in Chapter III.

2.4 The Robot Operating System

The Robot Operating System (ROS) is a flexible Linux-based framework for writing robotics software that provides a collection of tools and libraries to simplify the task of creating complex and robust robot behavior [19]. As a key feature of ROS is collaborative development, The ROS Wiki[1] is the primary resource for information on ROS distributions, features, tools, and packages. A detailed look at the specific ROS network setup used in this research is included in Chapter III, but a brief overview of the features of ROS relevant to this research is introduced in the following sections.

2.4.1 ROS Nodes and Topics

At its most basic level, ROS implements a system of *nodes* that communicate asynchronously with information in *messages* [19]. Messages are transmitted via unique *topics* that identify a specific message and its data type. Nodes can either *publish* or *subscribe* to a topic to send or receive the message data, respectively. A few particularly helpful ROS command-line tools for navigating and understanding the ROS nodes and messages of a network are shown in Table 1.

Table 1: Common ROS Commands

user@hostname$ **rosnode list**
user@hostname$ **rosnode info [node]**
user@hostname$ **rostopic list**
user@hostname$ **rostopic type [topic]**
user@hostname$ **rostopic echo [topic]**
user@hostname$ **rosrun rqt_graph rqt_graph**

[1]For documentation, tutorials, and the latest ROS distributions, visit http://wiki.ros.org/

As expected, **rosnode list** shows all of the active nodes on the network while **rosnode info** gives details of the node including the topics it subscribes to and publishes. Next, **rostopic list** outputs the topics in the same way as **rosnode list**, and **rostopic type** gives the message data type of the topic. The current actual message data of a topic can be output with **rostopic echo**. Lastly, **rosrun rqt_graph rqt_graph** produces a visual graph of the flow of data within the ROS network via its active nodes and topics. For example, a simple publisher node and subscriber node relationship is shown via rqt_graph in Figure 9.

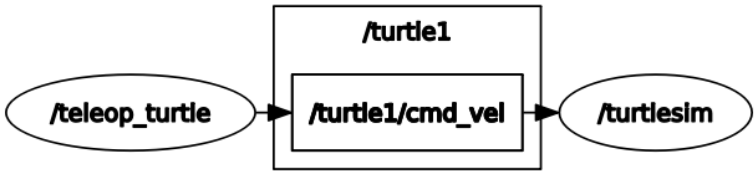


Figure 9: ROS rqt_graph

In Figure 9, the /teleop_turtle node is publishing the /turtle1/cmd_vel topic, and /turtlesim is subscribing to the topic. In this example, a turtle is being driven by a velocity command sent from a terminal running the teleoperation node. Turtlesim is a package included in the basic ROS tutorials and is further introduced in the following section on packages and services.

2.4.2 ROS Packages and Services

Software in ROS is organized into units called *packages*, which include the documentation, nodes, custom message definitions, and executable scripts required to implement the functionality of the package [19]. Package scripts can be run with **rosrun <package> <executable>**. In the turtlesim example introduced in Section 2.4.1, the simulation can be started by running **rosrun turtlesim turtlesim_node**, which initializes the simulator graphical user interface (GUI) along with its ROS nodes and topics as shown in Figure 10.

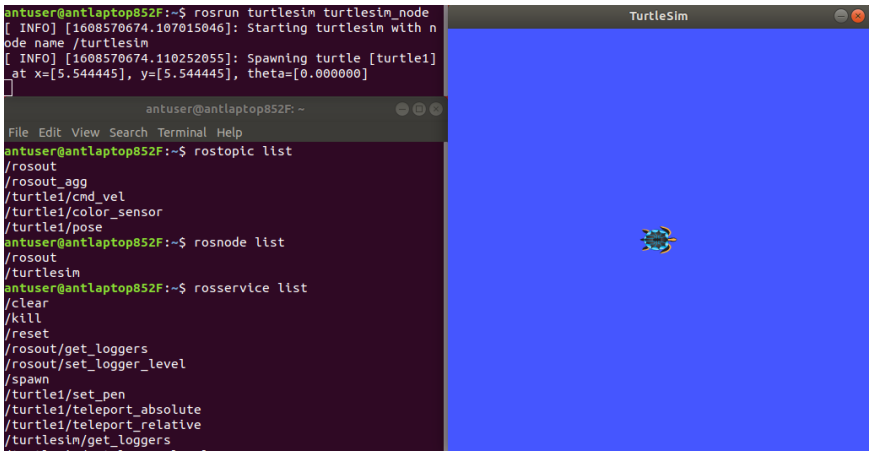


Figure 10: ROS Turtlesim Example

ROS packages can also include *services*, which implement behaviors that can be called over the ROS network. Services are primarily written in Python or C++ using the rospy or roscpp libraries. Available services on the ROS network can be listed with **rosservice list** and called with **rosservice call /<service-name> [service-args]**. In the turtlesim example, services are used to add or remove turtles from the simulation with **spawn** or **kill** and the simulation can be restarted with **reset**.

The wide range of packages developed by the ROS community that implement many commonly-used robotics tools and behaviors is one of the powerful features of ROS. The specific ROS packages and features used in this research are introduced in further detail in Chapter III.

2.5 Chapter Summary

This chapter introduces the main background concepts that enable the development of a 3-DOF mobile robotic hardware testbed for satellite proximity controls research. The demonstrated satellite behavior is based on the CW linearized equations of relative motion and is bound by the same assumptions as introduced in Section 2.1. The concepts of satellite dynamics of Section 2.1 and optimal control

methods of Section 2.3 are used in the design of the satellite control system in Chapter III. The following chapter also uses the mobile robot kinematics of Section 2.2 in the design of the robotic hardware and the robot operating system of Section 2.4 in the design of the testbed network.

III. Methodology

This chapter describes the testbed framework including the mobile robot hardware, the motion capture system, the Robot Operating System (ROS) network setup, and the control simulation. The intent of the framework is to provide a flexible platform for a user to test their own unique control scheme for their specific mission scenario. Section 3.1 introduces the hardware involved in the research, including the mobile robot and the motion capture system. Section 3.2 details the ROS network setup and computing environment that runs the testbed. Section 3.3 then describes the example satellite control simulation that is provided in-place of a user supplied controller for this research.

3.1 Hardware Systems

The two main hardware systems utilized in this research are the mobile robot and the motion capture system. The mobile robot is the omni wheel variant of the TurtleBot3 by Robotis[1] and the motion capture system is a set of eight VICON MX-F40 infrared cameras by Vicon Motion Systems[2].

3.1.1 Mobile Robot

The Turtlebot platform is marketed as a low-cost, personal robot kit with open-source software. Various versions have been developed since the original Turtlebot was released in 2010, as shown in Figure 11. The most recent release, the Turtlebot3, was released in 2017 with a Burger and a Waffle configuration. The Turtlebot3 Waffle originally included an Intel Joule, but was discontinued and replaced by a Raspberry Pi 3 with the Waffle Pi release. The Turtlebot3 Waffle Pi kit was used to build the

[1]For more information on the TurtleBot, see https://www.turtlebot.com/about/
[2]For more about Vicon Motion Systems, see https://www.vicon.com/

mobile robot platform in this research and includes the chassis components, servo motors, wheels, two on-board computers, and a lithium-ion polymer (LiPo) battery. The core components[3] of the standard Waffle Pi configuration are listed in Table 2 and are further detailed in the following sections.

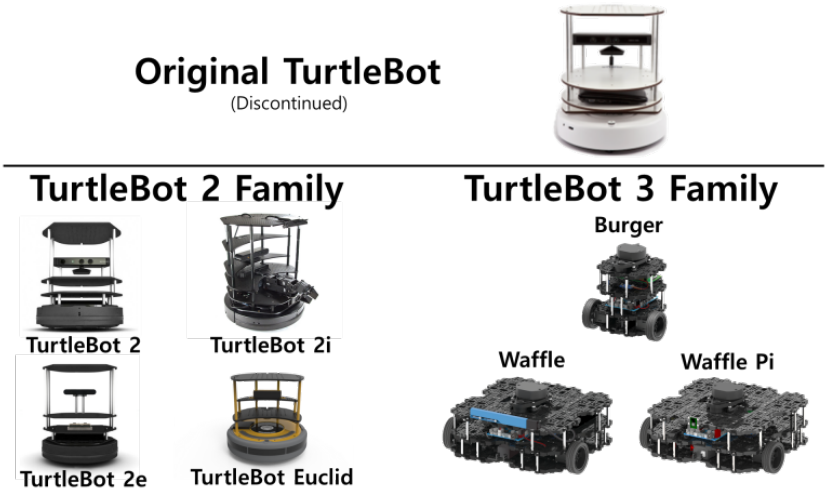


Figure 11: Turtlebot versions, from turtlebot.com

Table 2: Turtlebot3 Waffle Kit Core Components

Type	Item	Quantity
Chassis Component	Standard Waffle-Plate	24
Servo Motor	DYNAMIXEL XM420-W210-T	2
Wheels	Standard Circular Wheel + Tire	2
Wheels	Ball Caster	2
On-board Computer	Raspberry Pi 3 Model B+	1
On-board Computer	OpenCR1.0	1
Battery	LiPo 11.1V 1800 mAh	1

The Turtlebot3 documentation also includes 12 alternative configurations known as Turtlebot3 Friends[4] that can be assembled using the Waffle Pi kit and a few additional components. The standard Turtlebot3 Waffle configuration with two fixed

[3]See the parts list at https://emanual.robotis.com/docs/en/platform/turtlebot3/features/
[4]See the documentation at https://emanual.robotis.com/docs/en/platform/turtlebot3/locomotion/

standard wheels and two ball caster wheels is a nonholonomic robot, as discussed in Section 2.2. Therefore, the Waffle variant does not have the sufficient degrees of mobility to follow the required satellite trajectories in the constrained three degrees of freedom (3-DOF) scenario as presented in Section 2.1. Instead, the Turtlebot3 Omni variant from the list of Turtlebot3 Friends is used to provide a holonomic platform with three full degrees of mobility. The Turtlebot3 Omni, also referred to as the omnibot in this research, uses three omnidirectional Swedish wheels. The Swedish wheel constraints are introduced in Section 2.2.2, and the wheel is shown in Figure 12 below.

Figure 12: 60 mm Aluminum Swedish Wheel

3.1.1.1 Physical Components

A core component of the Turtlebot3 chassis is the standard waffle-plate. The waffle-plates are 127mm x 63mm x 9mm injected modeled plates and are designed to enable the chassis to be assembled[5] in a variety of different shapes. The Turtlebot3 Waffle consists of three layers of eight waffle-plates assembled as shown in Figure 13.

[5]See assembly instructions at https://emanual.robotis.com/docs/en/platform/turtlebot3/hardware_setup/

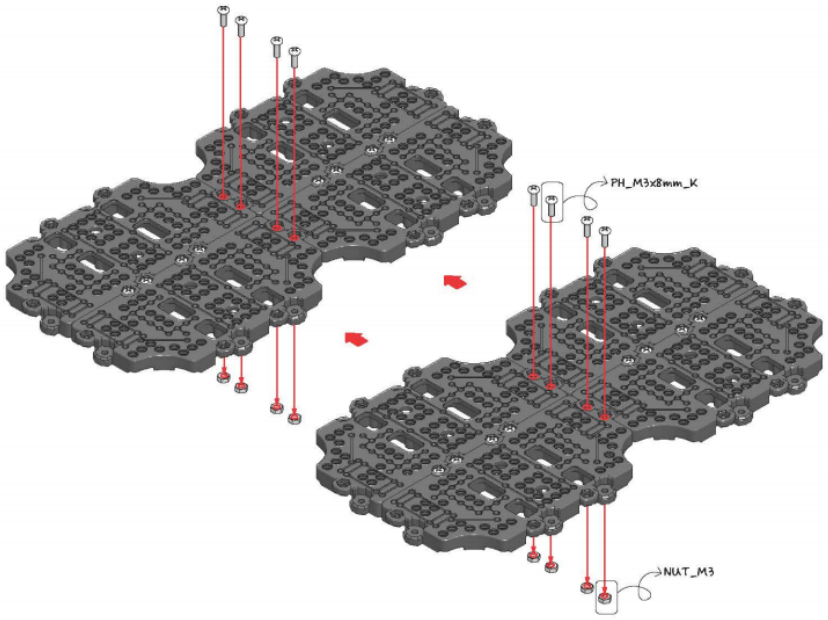


Figure 13: Waffle-plate Layer Assembly, from Robotis

The omnibot also uses the waffle-plates, but the layers are built from three waffle-plates attached around a central component. The Computer Aided Designed (CAD) model[6] for the central component is provided by Robotis and a 3D printed copy is shown in Figure 14.

Robotis also provided CAD models for the servo mounting brackets and the wheel adapters that were 3D printed locally. Three servo mounting brackets, shown in Figure 15, are installed on bottom layer of the omnibot to separate each of the Swedish wheels by 120 degrees and a distance of 115 mm from the center. The wheel adapters connect the 60mm Aluminum Swedish wheel with the DYNAMIXEL XM430 servo motor, as shown in Figure 16.

Fully assembled, the omnibot uses three layers of waffle-plates and measures at approximately 190 mm wide, 216 mm long, and 127 mm tall while weighing approximately 1.2 kg. The omnibot also includes retroreflective markers on the top layer that

[6]The CAD model is provided at http://www.robotis.com/service/download.php?no=684

Figure 14: Turtlebot3 Omni Layer

assist the motion capture system, which will be further introduced in Section 3.1.2. The assembled omnibot is shown in Figure 17.

3.1.1.2 On-board Computers

The omnibot, like the standard Turtlebot3 Waffle Pi, is controlled by two on-board computers: an OpenCR1.0 board and a Raspberry Pi 3 Model B+. The OpenCR board is powered by the LiPo battery and the Raspberry Pi is powered by a 5V output from the OpenCR. The OpenCR and the Raspberry Pi communicate via a micro-USB cable. Figure 18 shows the OpenCR, Raspberry Pi, and LiPo battery mounted to the first layer of the omnibot.

The Raspberry Pi runs Raspian, the Raspberry Pi Operating System (OS), which is a Debian Linux based OS. ROS is installed on the omnibot via Raspian and hosts a variety of Turtlebot nodes necessary for communication with the rest of the ROS network, which is further detailed in Section 3.2. The Raspberry Pi includes 2.4GHz and 5GHz wireless LAN as well as an Ethernet port for network connection. For de-

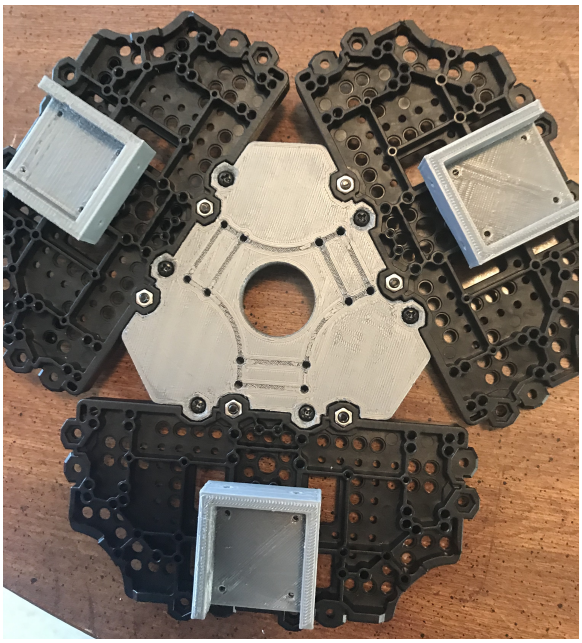


Figure 15: Omnibot Servo Mounting Brackets

Figure 16: Swedish Wheel Adapter

bugging the Raspberry Pi, Raspian can be accessed during operation directly via the on-board HDMI and USB ports or via Secure Shell (SSH). While the Raspberry Pi communicates with the ROS network, the OpenCR board controls the DYNAMIXEL servos via connection to each of the three TTL serial ports. The OpenCR also includes a three-axis gyroscope and three-axis accelerometer for on-board measurement.

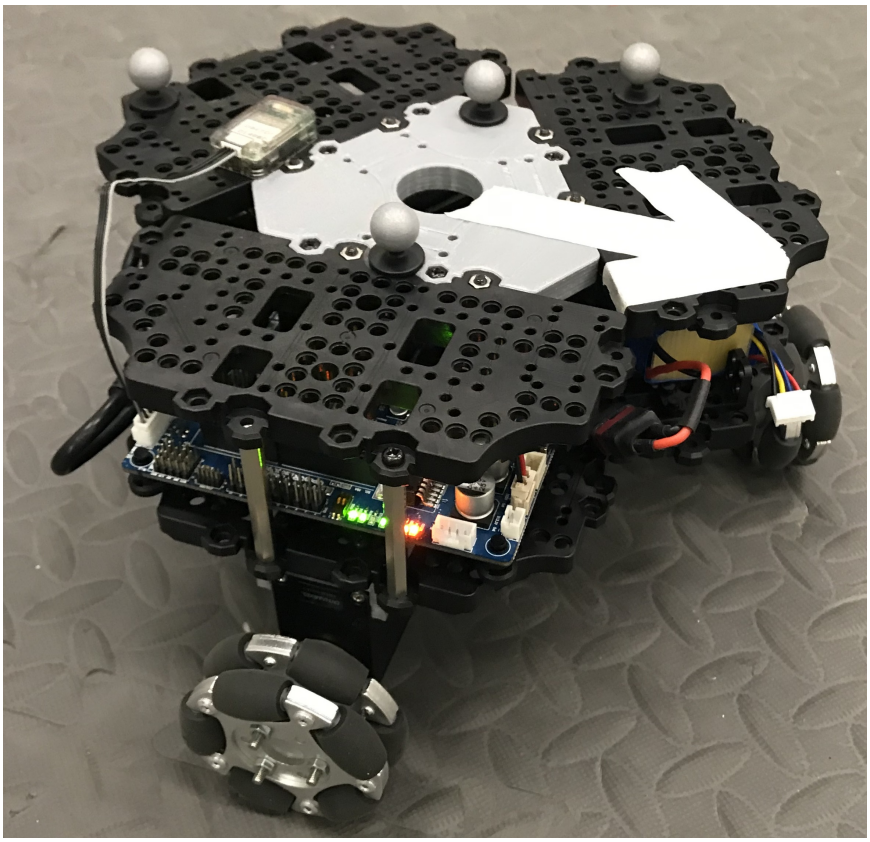


Figure 17: Assembled Omnibot

During debugging, the OpenCR can receive inputs from a remote controller via an attached BT-410 Bluetooth receiver to control the servos without the Raspberry Pi. OpenCR is compatible with the Arduino Integrated Development Environment (IDE) and implements the kinematics introduced in Section 2.2.4 to produce a commanded local velocity through individual wheel velocities.

3.1.1.3 Simulated Gazebo Model

Gazebo is an open-source robotics simulator included with ROS. A simulated model of a mobile robot is particularly useful during rapid control code iteration to discover errors before implementation on hardware. OpenBase[7] is an open-source omnidirectional mobile robot simulated via Gazebo shown in Figure 19. OpenBase

[7]OpenBase and its documentation are hosted at https://github.com/GuiRitter/OpenBase

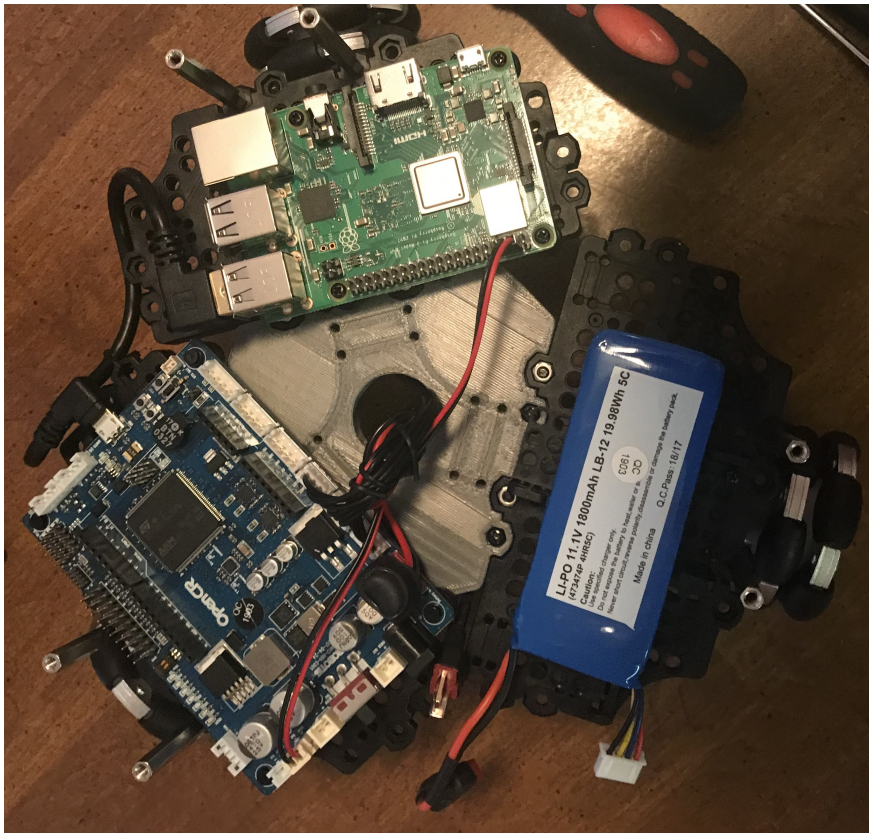


Figure 18: Raspberry Pi and OpenCR boards

can receive control commands via the same ROS topics as the omnibot hardware to easily swap between simulation and hardware. OpenBase was used throughout this research for prototyping and to continue development without access to the hardware in the ANT laboratory.

3.1.2 Motion Capture System

The ANT Lab is outfitted with a motion capture system consisting of eight VICON MX-F40 cameras by Vicon Motion Systems. The VICON system measures the position on the omnibot to an accuracy of up to one mm to provide true position data for the controller and for performance analysis. The cameras are arranged along the walls of the VICON chamber as shown in Figure 20.

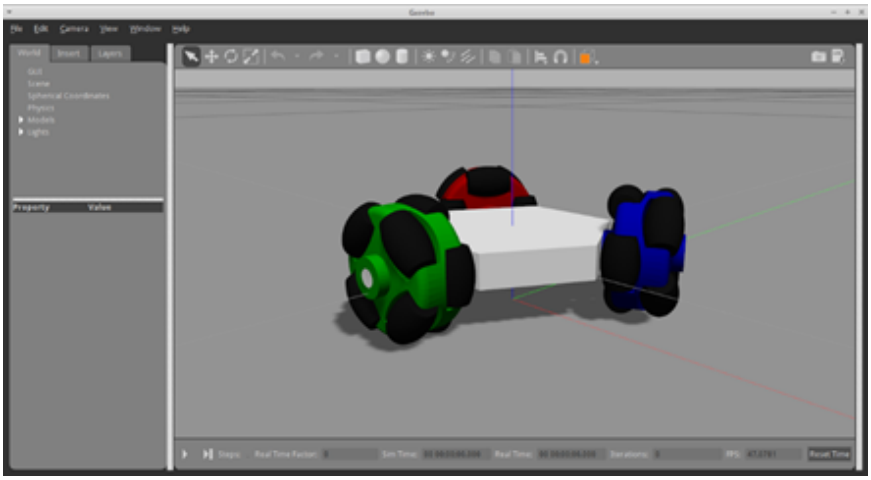


Figure 19: Gazebo Simulator with OpenBase, by GuiRitter

3.1.2.1 VICON Camera Setup

The VICON cameras are managed by the VICON Tracker software running on a Windows 10 Desktop PC in the ANT Laboratory. Once the cameras are calibrated, they are shown in the software matching their physical arrangement as displayed in Figure 21.

The VICON infrared cameras use passive retroreflective markers to measure position. The retroreflective markers are arranged in an asymmetrical pattern to define an object to be tracked as shown on the omnibot in Figure 17. Once defined and detected by the cameras, the object is displayed in the Tracker software as shown in Figure 22.

3.1.2.2 VICON Data

VICON data can be output on the ROS Network via the vicon_bridge package. This package creates a ROS node that broadcasts a geometry_msgs/TransformStamped message on the vicon/omnibot/omnibot topic. This message type includes the current time stamp, translation, and rotation of the object and is used to calculate the omnibot pose in the world frame as shown in Figure 23. The x and y values can be

Figure 20: ANT VICON Chamber

directly read from the translation data, but the θ component is calculated from the rotation data given in quaternions[8]. The MATLAB function quat2angle[9] is used for conversion in the controller used in this research, and it returns angles in the range of ± 180 degrees.

[8]See https://eater.net/quaternions for Sanderson's visualized explanation of quaternions
[9]See https://www.mathworks.com/help/aerotbx/ug/quat2angle.html

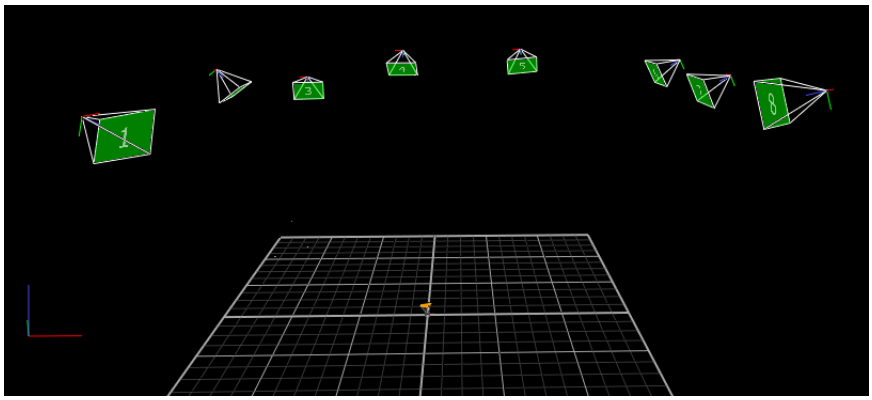


Figure 21: VICON Tracker Camera Arrangement

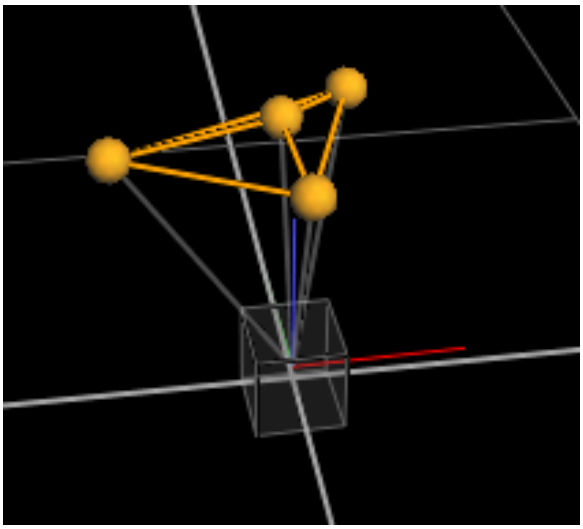


Figure 22: VICON Tracker Omnibot Object

Figure 23: VICON Example Pose Data

3.2 Testbed Network

The testbed network uses ROS to connect the ROS Master, the VICON system, the satellite control simulation, and the omnibot mobile robot platform. The ROS network enables the necessary data exchange between the four computers involved in the satellite control demonstration.

3.2.1 Network Overview

A NETGEAR Nighthawk R7000 Dual-Band WiFi router is used to host the testbed network. The 5Ghz WiFi band is used to connect the omnibot's Raspberry Pi to the network to enable untethered movement in the VICON chamber. Connection via an ethernet cable is also possible. The various equipment in the network require static IP addresses; therefore, the router is used without Dynamic Host Configuration Protocol (DHCP). The static IP address assignment with a subnet mask of 255.255.255.0 is given in Table 3.

Table 3: Static IP Address Assignment

Equipment	Static IP Address
VICON Camera Tracker System	192.168.10.1
Nighthawk WiFi Router	192.168.10.21
ROS Master on Ubuntu 18.04 Laptop	192.168.10.22
MATLAB on Windows Laptop	192.168.10.23
Turtlebot ROS Node on Raspberry Pi	192.168.10.24

3.2.2 Computing Environment

The following section details the function of each of the networked computers in their support of the operation of the omnibot. A distributed computing environment is used per the Turtlebot3 setup requiring a Linux-based PC hosting the ROS network and ROS Master separate from the Raspberry Pi. An alternative configuration to

consolidate more of the functionality on-board the omnibot could be achievable, but it would require a different control simulation or potentially an additional on-board computer and was not tested in this research.

3.2.2.1 ROS Master

The ROS Master and Parameter Server is hosted on a Dell Precision 7720 Laptop running Ubuntu 18.04 referred to as the Master Laptop. The Master Laptop is the central hub that connects the control simulation, the omnibot, and the VICON motion capture data. Figure 24 shows the three terminals that are active on the Master Laptop during hardware demonstration.

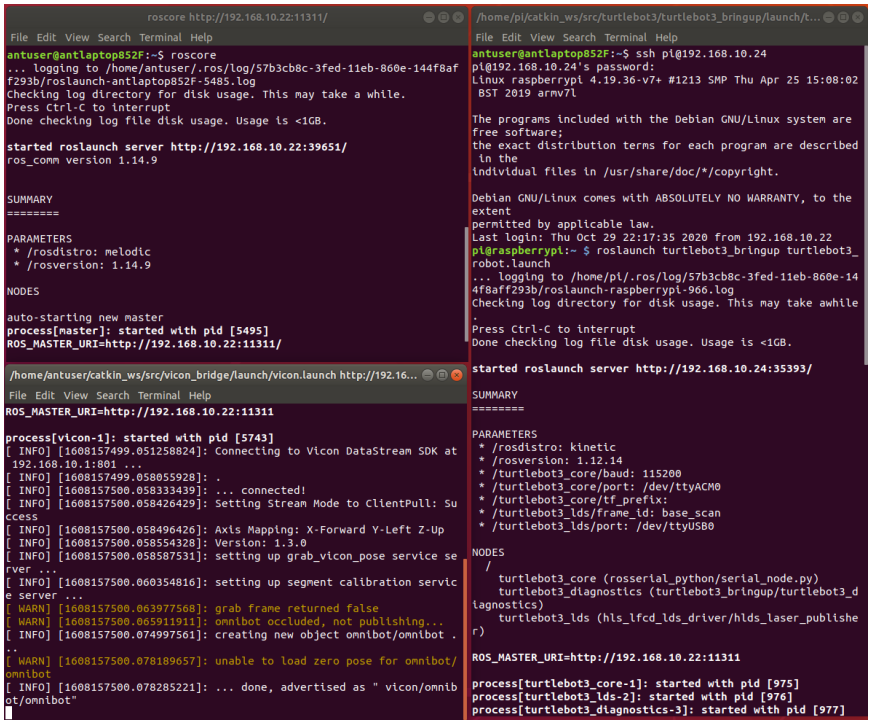


Figure 24: Master Laptop Active Terminals

The ROS Master is started with **roscore**, the VICON data is imported into the ROS network with the VICON bridge via **roslaunch vicon_bridge vicon.launch**, and an SSH connection is opened with the Raspberry Pi to initialize the Turtlebot3

package ROS nodes.

Another function of the Master Laptop is running the open_base simulation via Gazebo as introduced in Section 3.1.1.3. Gazebo is particularly useful for rapidly prototyping new robot behavior and code developments that do not require the full hardware and laboratory setup. While ROS can be run via a Linux-based virtual machine on a Windows PC, the Gazebo simulator performance is significantly degraded if hosted via a virtual machine. Even with the discrete NVIDIA Quadro P5000 Mobile graphics card of the Master Laptop, the Gazebo simulated omnidirectional robot model utilized in this research ran at a real time factor of approximately 0.5 to maintain stable performance.

3.2.2.2 VICON Tracker

The VICON Tracker software is installed on an ANT laboratory Windows 10 Desktop PC as introduced in Section 3.1.2. The real-time position of the omnibot is displayed via Tracker as shown via Figure 21. Hardware tests can also be recorded via Tracker to record the full position history over the duration of a test. A recording can then be replayed to the ROS network from Tracker to have the vicon_bridge node outputs replicate a previously performed test for further analysis.

3.2.2.3 MATLAB Control Simulation

The MATLAB satellite control simulation is run on a Microsoft Surface Pro 6 with Windows 10. MATLAB is used to host the control simulation due to its compatibility with the GPOPS-II optimal control solver and the MATLAB ROS Toolbox. The ROS Toolbox enables MATLAB to publish and subscribe to information on a ROS network, even via a Windows computer. A MATLAB approach was also chosen to enable further collaboration with others performing controls research via MATLAB

or Simulink. An alternative controller developed in Python or C++ could also be implemented as a ROS service without needing MATLAB, but that was not investigated in this research. The MATLAB controller receives timing, pose, and IMU data via subscription to ROS topics, and publishes a commanded velocity for the omnibot. The controller is designed to work with either simulated or real hardware, but the topic names for each setup are different, as shown in Table 4 below. Further details of the controller algorithm are given in Section 3.3.

Table 4: MATLAB Controller ROS Topics

	Simulated Hardware	Real Hardware
Timing	/clock	/vicon/omnibot/omnibot
Pose	/open_base/pose/world	/vicon/omnibot/omnibot
IMU	- (N/A)	/imu
Velocity Command	/cmd_vel	/cmd_vel

3.2.2.4 Raspberry Pi

While the general setup of the Raspberry Pi and connection to the ROS network was introduced in Section 3.1.1.2, this section will cover the software function and individual ROS nodes. Once a SSH connection is established with the Raspberry Pi, the TurtleBot3 node turtlebot3_core is initialized via **roslaunch turtlebot3_bringup turtlebot3_robot.launch**. The turtlebot3_core node subscribes to the published velocity command from the MATLAB controller via /cmd_vel, which is then distributed to the individual wheel commands via the connected OpenCR board. The turtlebot3_core node then publishes gyroscope and accelerometer measurements via /imu and additional sensors not used in this research via /sensor_state. The turtlebot3_core also publishes various diagnostic information via /battery_state and /diagnostics.

3.3 Satellite Control Simulation

In this section, a satellite control simulation is introduced that commands the mobile robot through an example satellite inspection scenario to demonstrate the capabilities of the hardware testbed. The control simulation consists of two parts, the trajectory generation via optimal control methods and the control algorithm implemented via MATLAB and ROS. The results of implementing this control simulation via both real and simulated hardware are presented in Chapter IV.

3.3.1 Example Satellite Servicing Scenario

Based on the background presented in Section 1.4 (assumptions), Section 2.1 (satellite dynamics), and Section 2.3 (optimal control), a cooperative satellite inspection scenario is developed that involves a single deputy satellite inspecting a rotating chief. The demonstrated component of the inspection involves the final approach of the deputy, where the deputy begins approximately 10 m from the chief. The goal of the scenario is for the deputy to reach an ideal distance and alignment to inspect a point of interest on the chief in minimum time.

The relative orbital motion of the chief and deputy satellites is given by the Clohessy-Wiltshire (CW) equations in the Local Vertical, Local Horizontal (LVLH) frame. The inspection scenario is a form of non-contact support and, therefore, does not include any contact dynamics. The scenario is then constrained by the 3-DOF capabilities of the mobile robot testbed, which includes two translational degrees of freedom (DOF) and two rotational DOF. The radial (x) and in-track (y) components of the relative orbital motion are investigated while the decoupled cross-track (z) motion is nullified. Rotation about the cross-track axis is implemented as the single rotational DOF of the testbed, while orientation about the radial and in-track axes is held constant.

The deputy satellite begins at an equilibrium point of the CW equations at relative rest, with an offset from the chief along the in-track axis and all other relative positions and velocities zero. The chief satellite remains at the origin of the LVLH frame throughout the scenario while rotating at a constant rate about the cross-track axis. The guidance of the deputy satellite is constrained by force and torque constraints of a single thruster and a reaction wheel, as well as a collision avoidance constraint formulated as a circle around the chief satellite. The goal of the scenario is then given by four terminal conditions. A desired final position is formulated as being within a circle around the chief satellite and also being within a cone that originates from a point of interest on the chief. The cone also rotates along with the chief at a constant rate. The deputy then ends at relative rest with zero velocity in all 3-DOF while also pointing towards the chief. The combination of terminal conditions creates an ideal state for the sensors in this scenario to complete the inspection.

3.3.2 Time-Optimal Reference Trajectories

To find a time-optimal guidance solution to the scenario presented in Section 3.3.1, an optimal control problem is formulated in Equations (46) to (55) as

$$J = t_f, \tag{46}$$

$$[x_0, y_0, \theta_0] = [0, y_e, \theta_e], \tag{47}$$

$$\left[\dot{x}_0, \dot{y}_0, \dot{\theta}_0\right] = [0, 0, 0], \tag{48}$$

$$F \in [0, F_{max}], \tag{49}$$

$$|\tau| \leq \tau_{max}, \tag{50}$$

$$R_{KOZ}^2 \leq (x)^2 + (y)^2, \tag{51}$$

$$R_G^2 \geq (x_f)^2 + (y_f)^2, \tag{52}$$

$$R_T^2 \geq (x_f - R_G \cos(\phi_f))^2 + (y_f - R_G \sin(\phi_f))^2, \tag{53}$$

$$|\arctan\left(\frac{y_f}{x_f}\right) - \theta_f| = \pi, \tag{54}$$

$$\left[\dot{x}_f, \dot{y}_f, \dot{\theta}_f\right] = [0, 0, 0], \tag{55}$$

where the objective function is given as Equation (46), the initial state is given as Equations (47) and (48), the control constraints are given as Equations (49) and (50), the collision avoidance constraint is given as Equation (51), and the terminal conditions are given as Equations (52) to (55).

3.3.2.1 Scenario Constants

A base configuration of the scenario constants from the CW dynamics in Equations (11) to (13) and optimal control constraints in Equations (46) to (55) is listed in Table 5 for the experiments included in the results of Chapter IV.

Table 5: Scenario Constants Base Configuration

Constant Name	Variable	Value
Chief Mean Motion	n	7.2921e-05 1/s
Deputy Mass	m	100 kg
Deputy Moment of Inertia	I_z	16.67 kg·m^2
Deputy equilibrium point (in-track)	y_e	10 m
Deputy equilibrium point (orientation)	θ_e	$\frac{\pi}{2}$ rad
Maximum Thrust Force	F_{max}	1 N
Maximum Torque	τ_{max}	1 N·m
Collision Avoidance Radius	R_{KOZ}	2 m
Goal Distance Radius	R_G	5 m
Cone Approximation Radius	R_T	1 m

3.3.2.2 GPOPS-II Settings

The commercial optimization solver GPOPS-II introduced in Section 2.3.3 will be used to solve the optimal control problem formulated in Section 3.3.2 for the time optimal trajectory. IPOPT was used as the nonlinear programming (NLP) solver. Default settings for GPOPS were used except for the mesh tolerance was tightened to 1e-6 and the maximum mesh iterations were increased to 20. A single-phase formulation in GPOPS was possible for this scenario, but a multi-phase approach could be used to demonstrate more complex scenarios with this same setup in future research.

3.3.3 Control Algorithm

The control algorithm in Algorithm 1 is used to command the mobile robot to follow the time-optimal reference trajectories found in Section 3.3.2. This process involves using waypoints along the trajectory to guide the mobile robot to the goal. However, it is not sufficient for the purpose of this research for the mobile robot to simply follow the positions along the trajectory according to its own limits with a response tuned to the specific hardware, as it would with a proportional-integral-

derivative (PID) or linear-quadratic regulator (LQR) based controller. Instead, it should replicate the kinematics of the deputy satellite as if it was in its orbital environment. Therefore, the waypoints of the trajectory are primarily based on time to ensure the motion throughout the trajectory remains synchronized with the timing of the optimal control solution.

Algorithm 1 Open Loop Controller

```
1: function CMDVELWPOL(hardware, vel_wp)
2:     for i ← 1, LENGTH(vel_wp) do                    ▷ Loop over waypoints
3:         ẋ_des, ẏ_des, θ̇_des ← vel_wp(i)             ▷ New waypoint velocity
4:         while time < vel_wp(i, 1) do                ▷ Until next waypoint
5:             time, x, y, θ ← VICON_BRIDGE            ▷ Get current pose
6:             ẋ, ẏ, θ̇ ← DIRTY_DERIVATIVE              ▷ Calculate current velocity
7:             vel_cmd ← COORD_TRANSFORM                ▷ To local robot reference frame
8:             ROS_PUBLISH(vel_cmd)                     ▷ Send velocity command
9:         end while
10:    end for
11: end function
```

The time and velocity components of the trajectory from the GPOPS solution is stored in the r-by-4 vector *vel_wp*, where r is the number of collocation points from the GPOPS solution. Column 1 of the vector is the timing of the waypoint, where Columns 2 through 4 are the x, y, and theta components of the deputy's velocity at each waypoint as shown in Equation (56). During each iteration of the control loop commanding the mobile robot to the next waypoint, navigation is performed via data from the motion capture system and velocity commands are processed and sent to the mobile robot.

$$vel_wp = \begin{bmatrix} t_0 & \dot{x}_0 & \dot{y}_0 & \dot{\theta}_0 \\ \vdots & \vdots & \vdots & \vdots \\ t_f & \dot{x}_f & \dot{y}_f & \dot{\theta}_f \end{bmatrix} \tag{56}$$

3.3.3.1 Motion Capture Navigation

The VICON motion capture system introduced in Section 3.1.2 is used to perform the navigation component of the demonstration. The world frame position information is measured via the VICON_BRIDGE function on Line 5 of Algorithm 1. The measured pose throughout the demonstration is then compared to the optimal control solution to analyze the performance of the testbed.

Although the VICON_BRIDGE only provides the position information, the mobile robot velocity at each measurement can be calculated via a band-limited, or dirty derivative [20]. The discrete time dirty derivative used on Line 6 of Algorithm 1 is given as

$$u_D[n] = \left(\frac{2\sigma - T_s}{2\sigma + T_s}\right) u_D[n-1] + \left(\frac{2}{2\sigma + T_s}\right) (e[n] - e[n-1]) \tag{57}$$

where σ is small, and $\frac{1}{\sigma}$ is the bandwidth of the differentiator in radians per second [20]. In Equation (57), n is the the sample, T_s is the sample period, and in this research the error signal, e, is used as the difference between the current and previous position measurements. The calculated velocities are then compared to the optimal control solution velocities throughout the trajectory.

3.3.3.2 Open Loop Velocity Commands

In the open loop method of Algorithm 1, desired world frame velocities for each waypoint are taken from *vel_wp* as shown in Equation (56). The world frame velocities are then converted to the mobile robot local frame using the orthogonal rotation matrix shown in Equation (22) via COORD_TRANSFORM on Line 7 of Algorithm 1. The local frame velocities in *vel_cmd* are then published to the */vel_cmd* topic on the ROS network using ROS_PUBLISH on Line 8 to command the mobile robot via its

on-board computers as mentioned in Section 3.2.2.4.

3.3.3.3 Closed Loop Velocity Commands

An alternative closed loop method is shown in Algorithm 2, where the pose information of each waypoint is included via the r-by-4 vector *pose_wp* in Equation (58), which shares the same dimensions as the *vel_wp* vector.

$$pose_wp = \begin{bmatrix} t_0 & x_0 & y_0 & \theta_0 \\ \vdots & \vdots & \vdots & \vdots \\ t_f & x_f & y_f & \theta_f \end{bmatrix} \tag{58}$$

Algorithm 2 Closed Loop Controller

```
1:  function CMDVELWPCL(hardware, vel_wp, pose_wp)
2:      for i ← 1, LENGTH(vel_wp) do                    ▷ Loop over waypoints
3:          x_des, y_des, θ_des ← pose_wp(i)            ▷ New waypoint pose
4:          ẋ_des, ẏ_des, θ̇_des ← vel_wp(i)             ▷ New waypoint velocity
5:          while time < vel_wp(i, 1) do                ▷ Until next waypoint
6:              time, x, y, θ ← VICON_BRIDGE            ▷ Get current pose
7:              ẋ, ẏ, θ̇ ← DIRTY_DERIVATIVE              ▷ Calculate current velocity
8:              vel_correction ← CALC_FEEDBACK          ▷ Feedback error signal
9:              vel_cmd ← COORD_TRANSFORM               ▷ To local robot reference frame
10:             ROS_PUBLISH(vel_cmd)                    ▷ Send velocity command
11:         end while
12:     end for
13: end function
```

Algorithm 2 then uses the CALC_FEEDBACK function in Line 8 to calculate the feedback error signal with a linear interpolation

$$P_C = P_A + \left(\frac{t_C - t_A}{t_B - t_A}\right)(P_B - P_A) \tag{59}$$

where the expected current position, P_C, is estimated based on the time elapsed between the previous and next waypoints. In Equation (59), P_A and t_A are the pose

and time of the previous waypoint, P_B and t_B are the pose and time of the next waypoint, and t_C is the current time of the demonstration. The *vel_correction* vector is then calculated based on the error between the expected and actual position and added to the *vel_cmd* vector to correct for that error.

3.4 Chapter Summary

This chapter presents the overall framework of the 3-DOF satellite proximity operations hardware testbed. The simulated and real omnibot mobile robot hardware configurations are presented in Section 3.1.1, and results from tests of each are compared in Chapter IV. Section 3.2 includes an overview of the four computers that exchange data over the ROS network, including the computers handling the ROS Master, the VICON tracker software, the MATLAB control simulation, and the Turtlebot Nodes. The satellite control simulation of Section 3.3 introduces the optimal control formulation for the guidance trajectory generation, the implementation of the motion capture navigation, and the algorithms for both open and closed loop controllers. The results of testing the open and closed loop controllers on variations of the example satellite servicing scenario of Section 3.3.1 are analyzed in Chapter IV.

IV. Results and Analysis

This chapter presents the performance of the omnibot demonstrating a component of a satellite servicing inspection scenario. The background and methodology for the specific demonstration scenario is given in detail in Sections 2.1 and 3.3. A narrative description of the scenario is provided in Section 3.3.1. The results of demonstrating seven reference trajectories with varying chief initial conditions, state dynamics, and deputy physical characteristics are compared using simulated and real hardware, as well as using open and closed loop controllers. The results are presented to showcase the capabilities of using the testbed to analyze the behavior of a controller in a given example satellite servicing scenario.

4.1 Scenario Map Overview

While a narrative description of the test scenario with initial conditions, constraints, and terminal conditions is presented in Section 3.3.1, a graphical depiction of the scenario is shown in Figure 25.

The scenario map of Figure 25 represents the Local Vertical, Local Horizontal (LVLH) frame with the chief satellite designated by * at the origin. The collision avoidance constraint, or keep out zone (KOZ), is shown by the red circle around the chief with a radius of 1 m. The blue circle around the chief with a radius of 5 m is the goal distance for the deputy satellite. The region between the two concentric circles is a component of the terminal state. The other position component of the terminal state is to end within the cone designated by the two dark blue dashed lines coming from the chief. The green circle is used to approximate a target within the cone that is centered on the goal distance to simplify the constraints. Therefore, the final terminal position constraint is being within both the goal and cone target circles

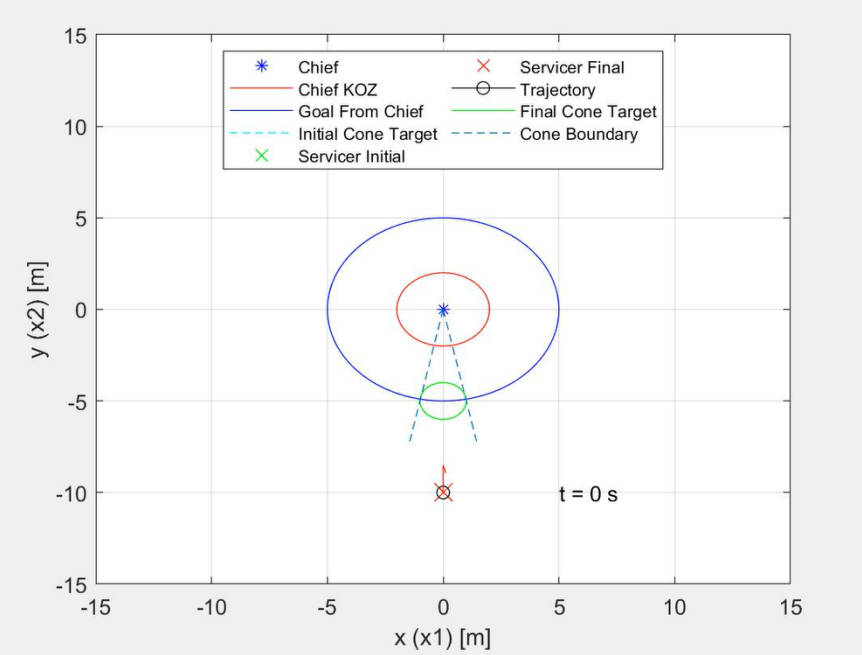


Figure 25: Scenario Map Initial

simultaneously. The deputy is represented as an X with a red arrow designating the orientation of the satellite and direction of thrust.

Throughout the trajectory, the collocation points from the GPOPS optimal control solver are represented by black circles, and the orientation of the deputy at each collocation point is represented by a magenta arrow, as shown in Figure 26. As the chief rotates at a constant rate in most of the test scenarios, a dashed circle is placed at the initial location of the cone target for reference, in cyan. The initial position of the deputy at the equilibrium point of the CW equations is also marked by a green X for reference.

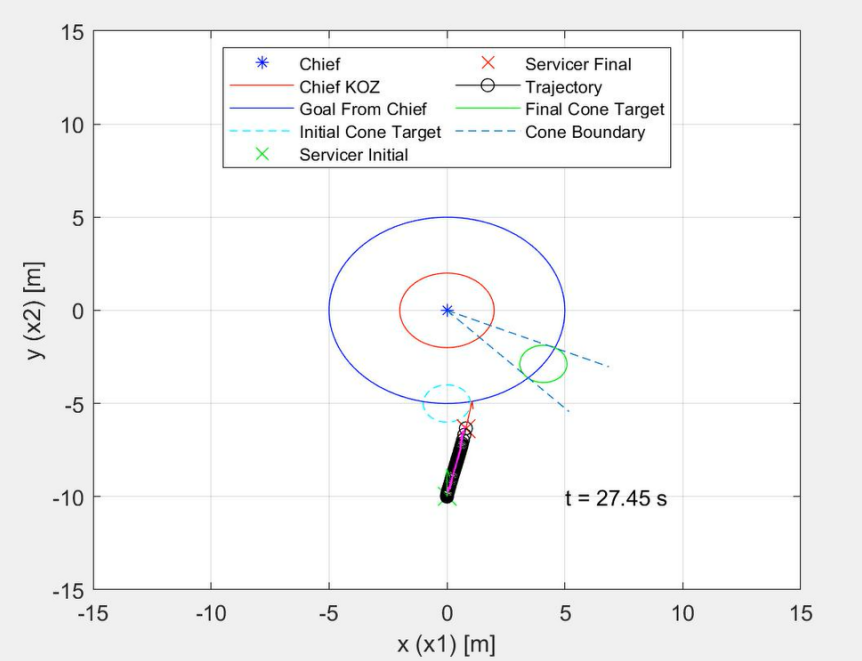


Figure 26: Scenario Map in Progress

4.2 Test Cases

This section analyzes the results of seven unique test cases that are formed from varying initial conditions of the optimal control problem in Section 3.3.2. The base configuration of the constants used in the tests cases is found in Table 5 of Section 3.3.2.1. The default for each test is the base configuration unless otherwise mentioned. While the scenario maps depict an area of 15 x 15 meters, the Autonomy and Navigation Technology (ANT) Center VICON chamber is not that large, so a scaling factor is introduced for the hardware tests that scales the optimal control solutions down by a factor to fit the smaller space and scales the results back up for comparison to the original trajectory. A one-tenth scaling factor is used for the hardware demonstrations unless otherwise mentioned.

Test Case 1 involves a stationary chief that remains at the origin of the LVLH frame without any rotation. Test Cases 2 and 3 introduce a constant rotation rate of 2 degrees per second for the chief. Test Case 2 has the chief starting at an initial

Table 6: Seven Test Cases

Case	Chief Initial	Chief Rate	Dynamics	Mass
1	$-\pi/2$ rad	0 deg/s	CW	100 kg
2	$-\pi/2$ rad	2 deg/s	CW	100 kg
3	0 rad	2 deg/s	CW	100 kg
4	$-\pi/2$ rad	2 deg/s	DI	100 kg
5	0 rad	2 deg/s	DI	100 kg
6	0 rad	2 deg/s	CW	200 kg
7	0 rad	2 deg/s	CW	50 kg

orientation of $-\pi/2$ radians, while the chief in Test Case 3 starts with an orientation of 0 radians. Test Cases 4 and 5 then use the same initial conditions as Test Cases 2 and 3 but use double integrator dynamics instead of the default state dynamics given by the Clohessy-Wiltshire (CW) equations of motion. The difference in trajectories between Test Cases 2 and 4 highlights the effect of the alternate dynamics. Finally, Test Cases 6 and 7 show the ability of the testbed to demonstrate the motion of a deputy satellite with different physical characteristics, such as mass.

For brevity, figures are provided within the following sections only when demonstrating unique concepts or behaviors of the testbed. Therefore, the bulk of the analysis is concentrated in the first three test cases, discussed in Sections 4.2.1 to 4.2.3. However, all output figures for all seven test cases are included for reference in Appendices A through G.

4.2.1 Test Case 1: Stationary Chief

The first demonstration trajectory is generated using a stationary chief with an initial orientation of $-\pi/2$ radians. In this scenario, the cone target is directly in front of the deputy. This simple case is presented to highlight the common behaviors of the GPOPS solutions, the robot simulation, and the actual hardware that are common throughout the more complex trajectories. The deputy primarily has to accelerate

forward along the radial axis, rotate to reverse the direction of thrust, and then decelerate to achieve the terminal position. The time-optimal trajectory generated by GPOPS is shown in Figure 27.

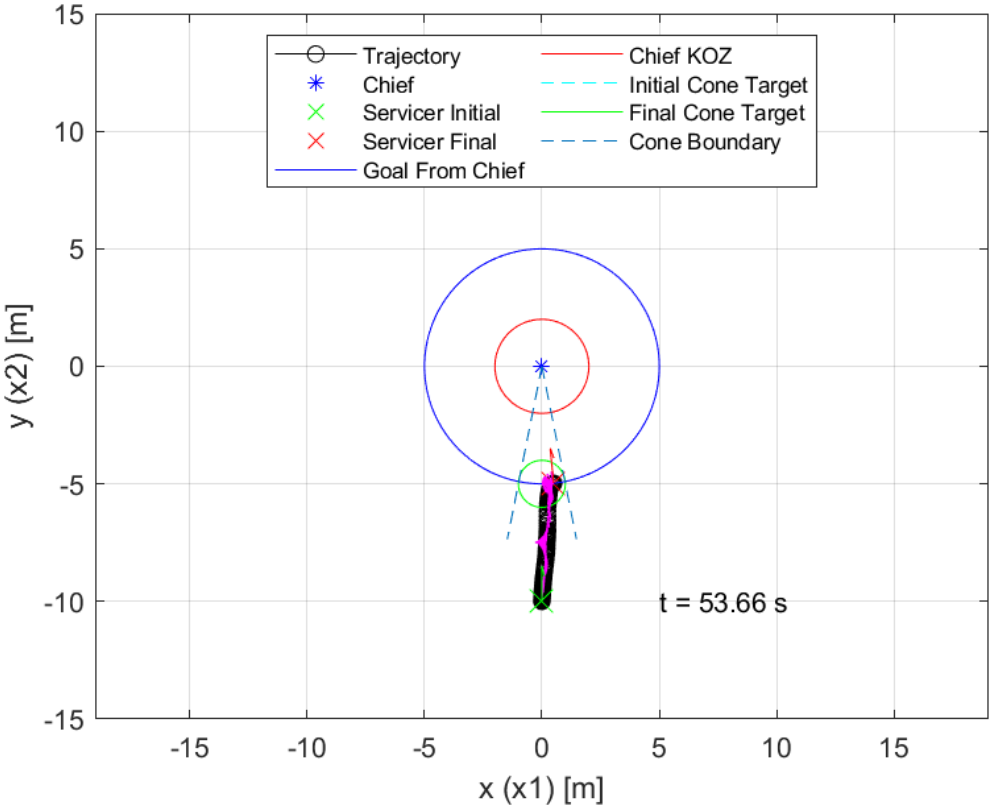


Figure 27: Test Case 1 Optimal Solution: Trajectory

However, since the deputy satellite only has one thruster fixed to its back, the time-optimal trajectory involves maximum thrust towards the target for the first half of the trajectory and then turning and thrusting away from the target final position with maximum thrust to reach relative rest with respect to the chief. The following control plots in Figure 28 show the thrust and torque commands for the deputy.

This behavior of applying maximum positive thrust followed by maximum negative thrust is a common behavior of time-optimal control due to Pontryagin's Maximum (or Minimum) Principle [16]. This type of control is also referred to as bang-bang

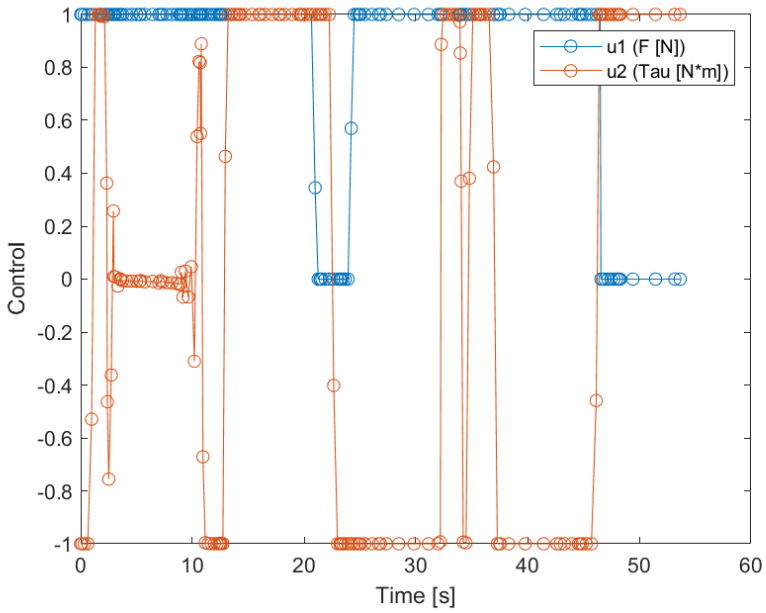


Figure 28: Test 1 Optimal Solution: Control

control due to the abrupt change between maximum applications of force, with this specific case representing a bang-off-bang pattern due to the pause occurring around 20 seconds into the trajectory [16]. The pause between the two applications of maximum thrust is due to the rotation of the deputy as it reorients to decelerate. Evidence of bang-bang control is also found in the state velocities, as shown in Figure 29.

As the terminal position is primarily offset from the initial position along the in-track (y) axis, the thrust is mainly applied when the deputy is aligned with the in-track axis. This results in the inverted V-shaped curve shown in Figure 29 where the in-track velocity (Vy) is rising and falling at a constant rate due to the constant acceleration from maximum force. The flat top of the Vy curve aligns with the pause in thrust during rotation. The full set of figures for the optimal solution is shown in Figure 30

Test Case 1 is then implemented on an omnidirectional mobile robot in simulation using open_base as discussed in Section 3.1.1.3. Figure 31 shows the results of testing

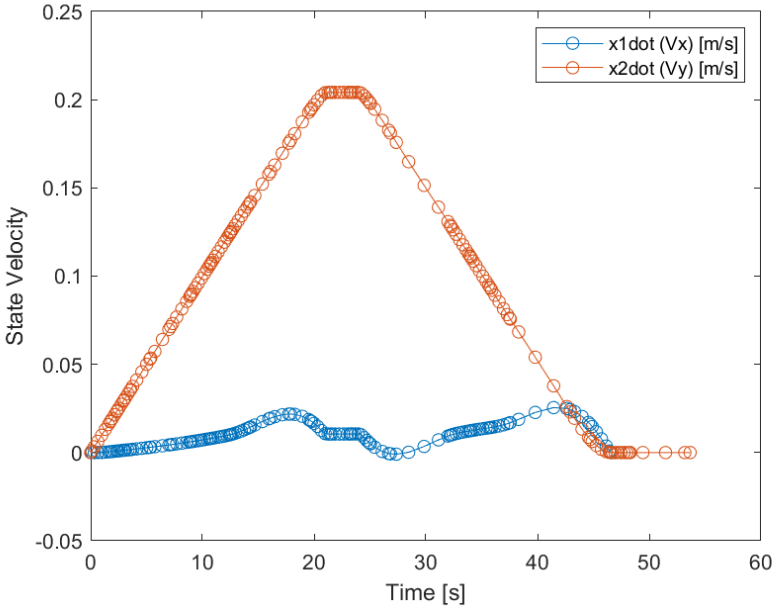
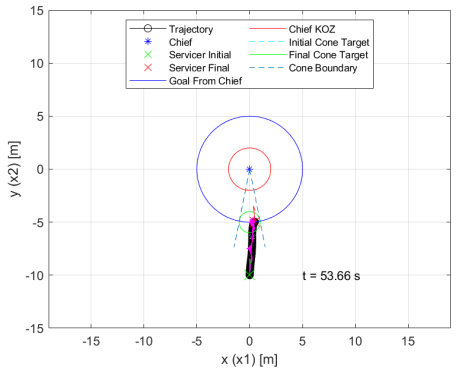
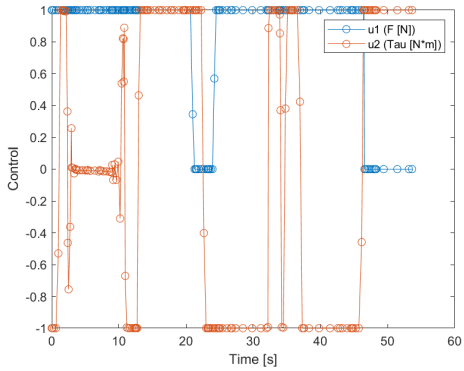


Figure 29: Test 1 Optimal Solution: State Velocities

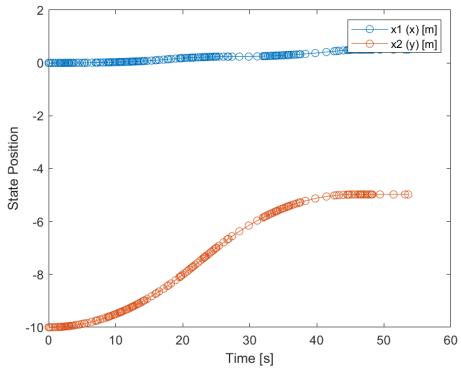
the open loop controller in Algorithm 1 of Section 3.3.3.2. While the simulated robot finishes close to the terminal position and at relative rest, the orientation error is significant. Figure 31c shows the x position error, Figure 31e shows the y position error, and Figure 31g shows the theta orientation error.



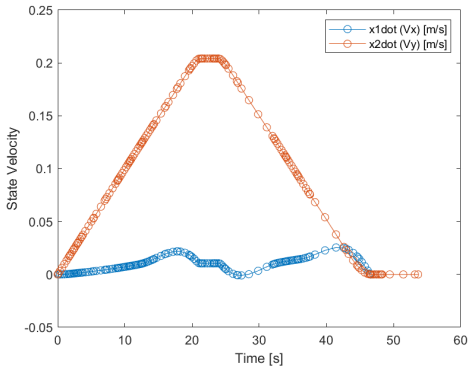
(a) Trajectory



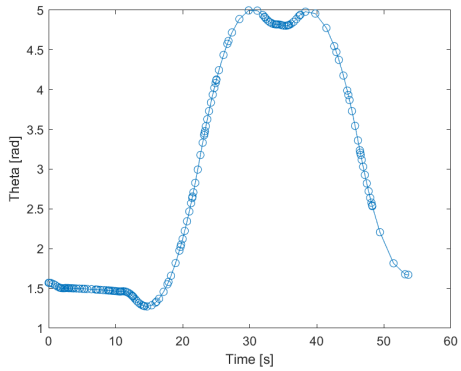
(b) Control History



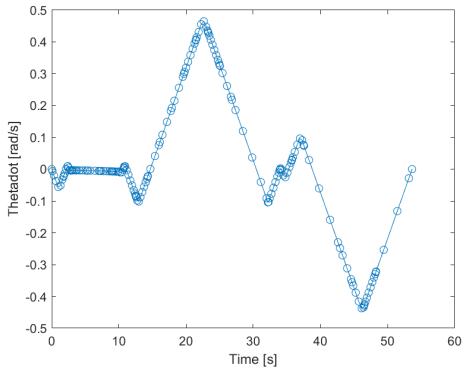
(c) Position



(d) Planar Velocities

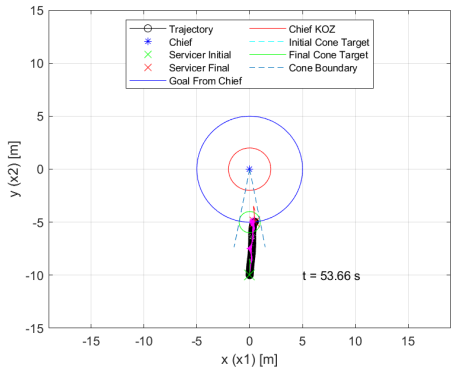


(e) Orientation

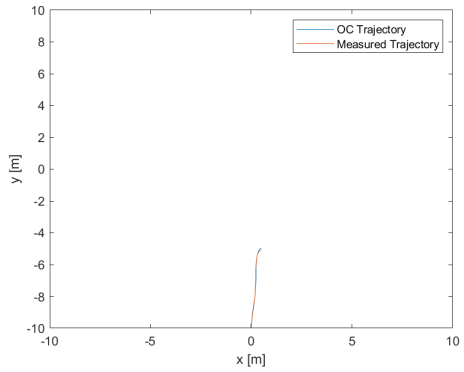


(f) Angular Velocity

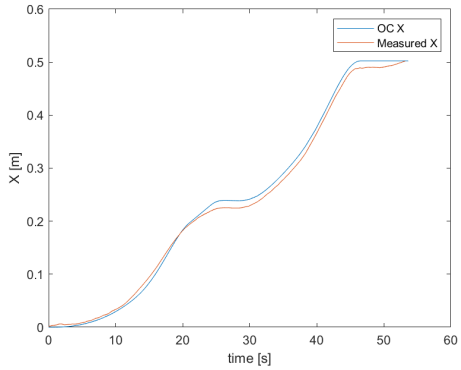
Figure 30: Test 1 time optimal solution, as computed using the GPOPS solver.



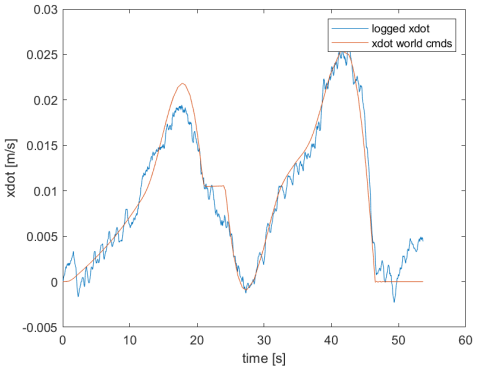
(a) Optimal Trajectory



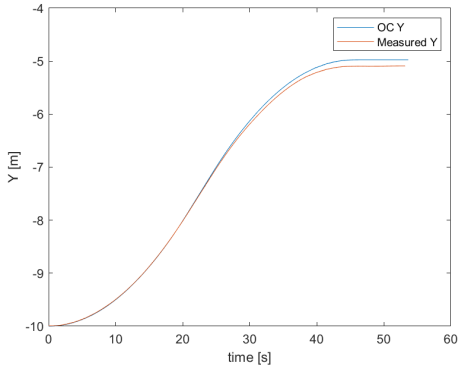
(b) Compared Trajectory



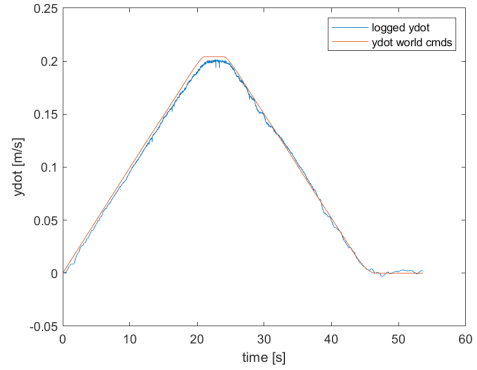
(c) Compared X



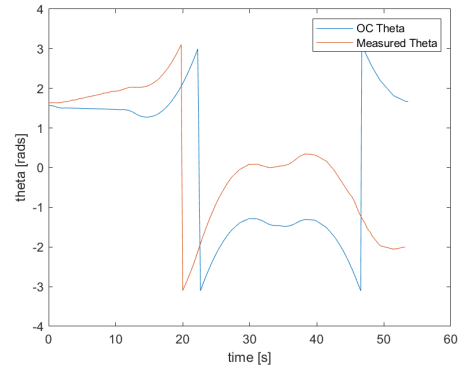
(d) Compared X Velocity



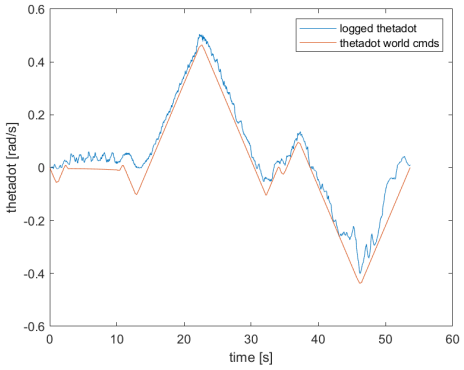
(e) Compared Y



(f) Compared Y Velocity



(g) Compared Theta



(h) Compared Theta Velocity

Figure 31: Test 1 simulated open loop control compared to the optimal control trajectory in Figure 30

While a large contributing factor for the orientation error is due to an error[1] with θ in the current version of the open_base simulation, it highlights the shortcomings of the open loop controller. Any error introduced throughout the trajectory accumulates and is not corrected. The state velocities are also shown in Figures 31d, 31f and 31h. It is important to note that the scale of the x values for position and velocity are almost one-tenth of the y values due to the limited motion in the x-axis for Test Case 1.

As expected, the y velocity demonstrates the inverted V-shape from the bang-bang control of the optimal control solution. Despite the open loop nature of the controller, the measured x and y trajectories track the optimal control solution fairly well with a root-mean-square error (RMSE) of 8.9 mm in x and 58.3 mm in y. However, the theta velocities are further off and the accumulated errors yield a RMSE over 1 radian (58.01 degrees) and can be seen in Figure 31g.

The same trajectory and open loop controller is then demonstrated on the omnibot hardware and shown in Figure 33. A sequence of four frames of the video from this demonstration is shown in Figure 32 to demonstrate the omnibot motion.

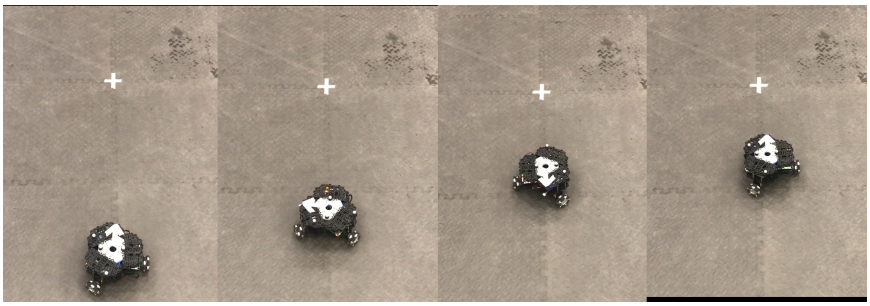
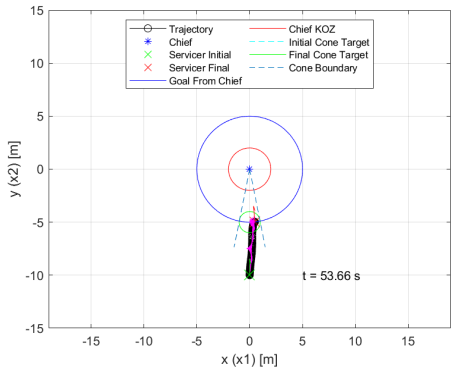
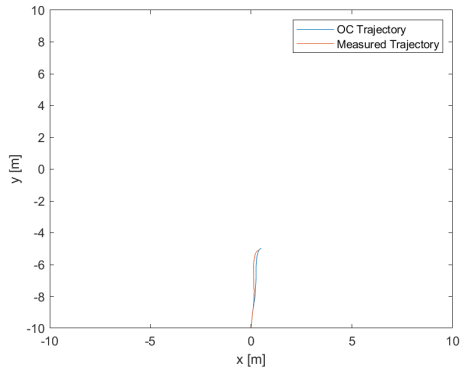


Figure 32: Test 1 Hardware Video Frames

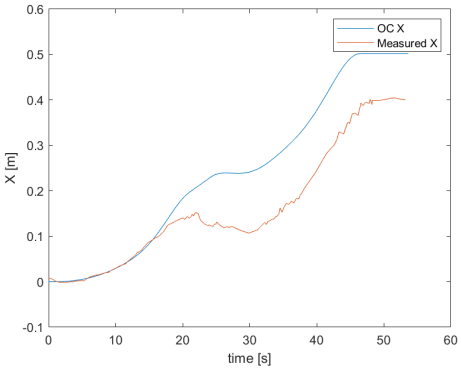
[1]See https://github.com/GuiRitter/OpenBase/issues/2 and https://github.com/GuiRitter/OpenBase/issues/7



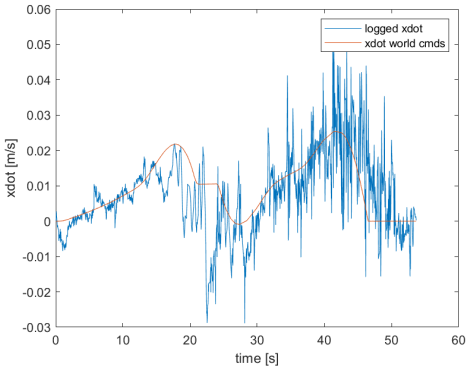
(a) Optimal Trajectory



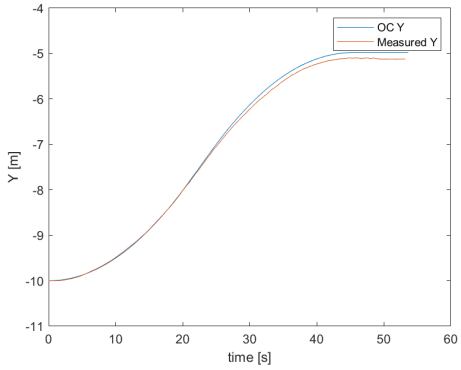
(b) Compared Trajectory



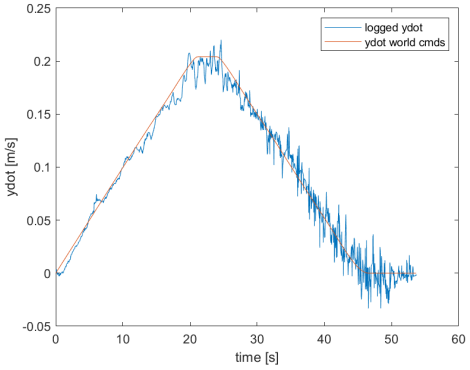
(c) Compared X



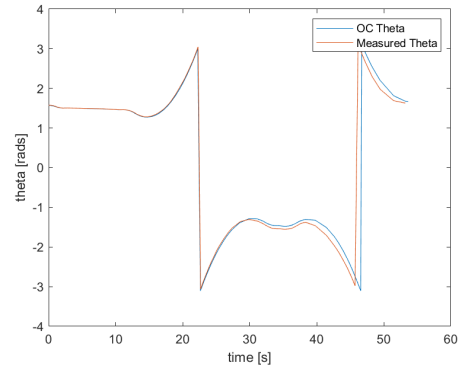
(d) Compared X Velocity



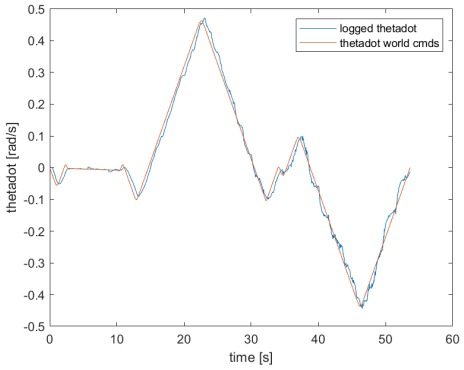
(e) Compared Y



(f) Compared Y Velocity



(g) Compared Theta



(h) Compared Theta Velocity

Figure 33: Test 1 hardware open loop control with data collected off the omnibot robot in the ANT motion capture lab, compared to the optimal trajectory shown in Figure 30

The pose of the omnibot over the trajectory is shown in Figures 33c, 33e and 33g. While the theta error is less than the error shown in Figure 31g for the simulated robot (0.095 rads vs 1.012 rads), the x error is much greater than the error in Figure 31c (83.9 mm vs 8.9 mm) and the y error is slightly greater than the error in Figure 31e (71.4 mm vs 58.3 mm). The need for feedback control is made clear to negate the accumulated error. Lastly, the state velocity outputs are shown in Figures 33d, 33f and 33h. Due to the one tenth scaling to fit the scenario within the dimensions of the ANT Center VICON chamber, the dirty derivative is estimating x axis velocities at the millimeter order of magnitude and generates the noisy measurement seen in Figure 33d.

4.2.2 Test Case 2: Rotating Chief 1

The second test case uses the same initial conditions as Test Case 1 in Section 4.2.1, but with the chief rotating at a constant two degrees per second (approximately 0.035 rad/s). The time optimal solution for this scenario is shown in Figure 35. One significant difference from Test Case 1 is that the trajectory covers more distance over a longer period of time (79.47 s vs 53.66 s) due to the cone target rotating to the opposite side of the initial position. The control history in Figure 35b is also more complex due to the deputy passing by the keep out zone constraint as close as possible without crossing into it. This test case is further compared to Test Case 4 in Section 4.2.4 where alternative dynamics are used instead of CW dynamics in the optimal control formulation.

The results from demonstrating the optimal solution in Figure 35 on the omnibot hardware with the open loop controller are shown in Figure 36. Similar to Test Case 1, error is accumulated throughout the trajectory yielding a RMSE of 123.6 mm in x, 250.8 mm in y, and 0.0737 rad in θ.

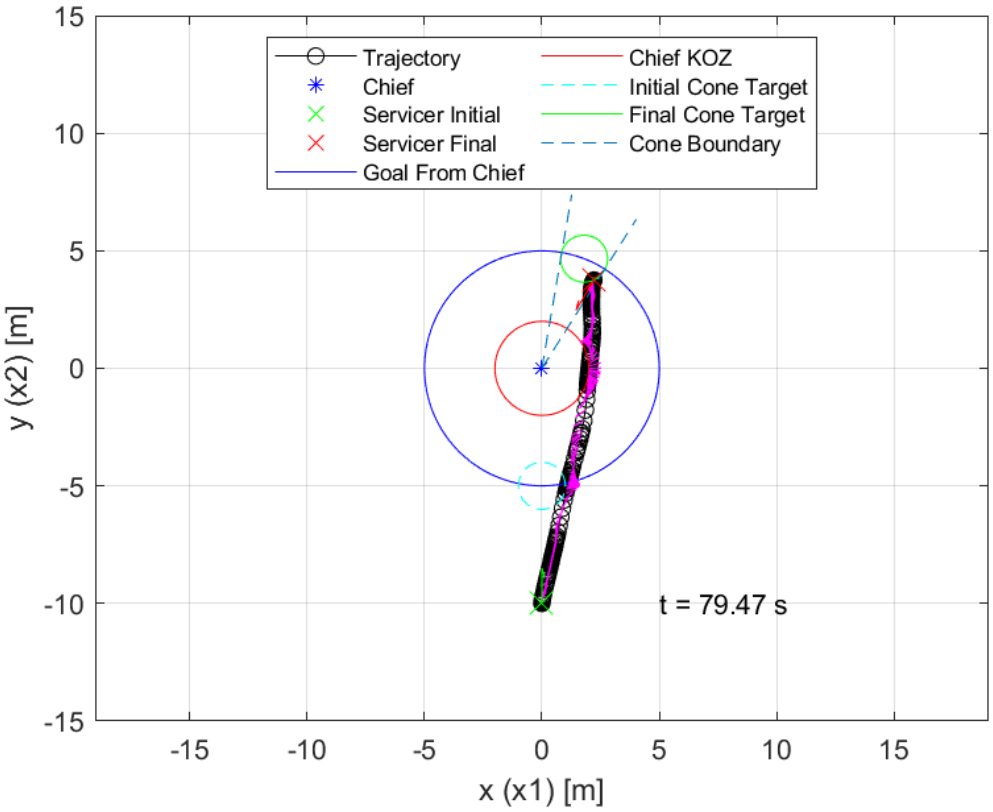
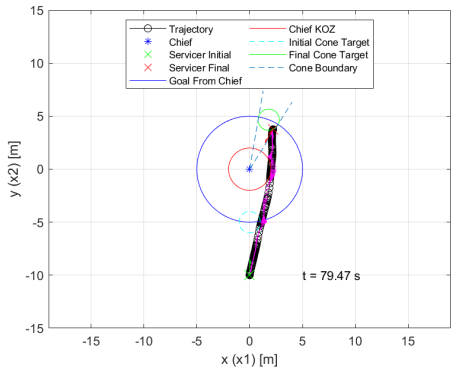
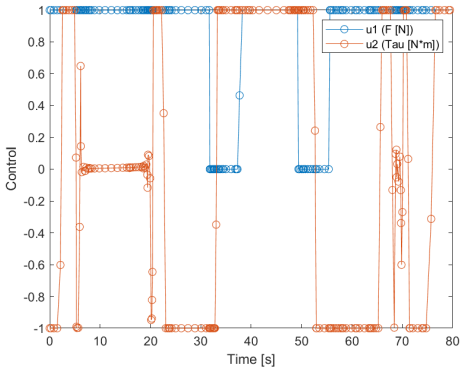


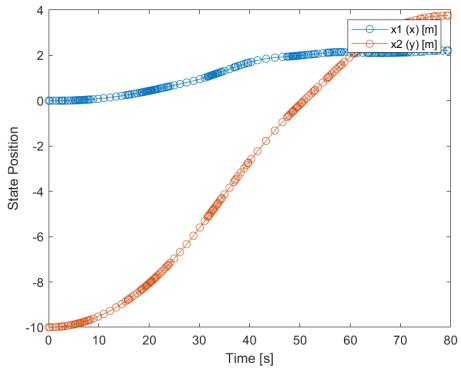
Figure 34: Test Case 2 Optimal Solution: Trajectory



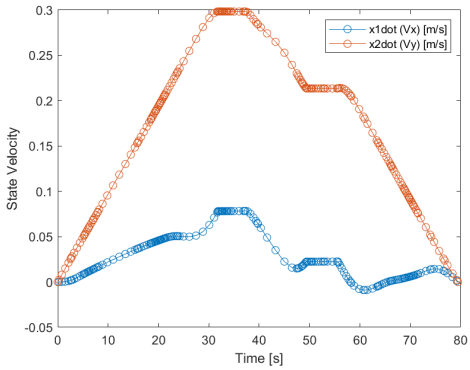
(a) Trajectory



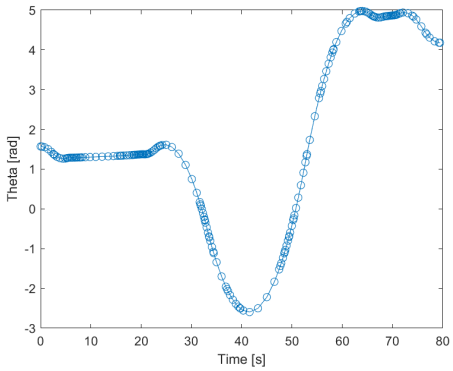
(b) Control History



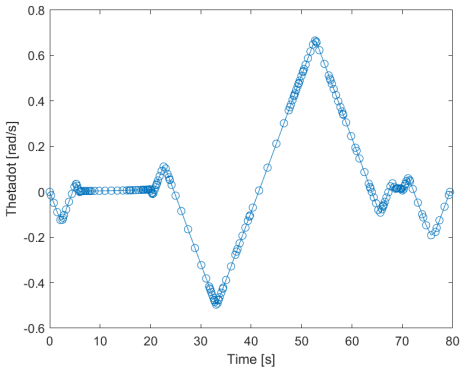
(c) Position



(d) Planar Velocities

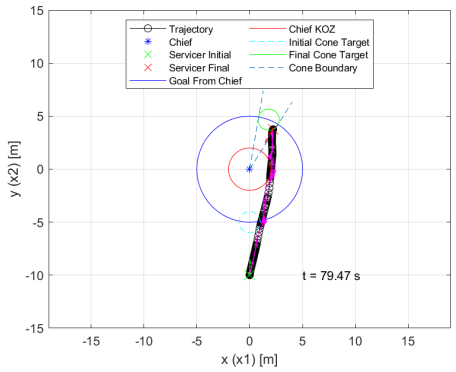


(e) Orientation

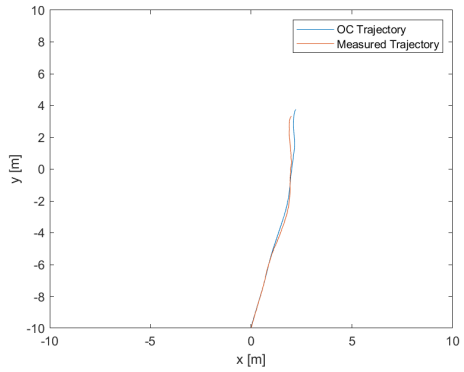


(f) Angular Velocity

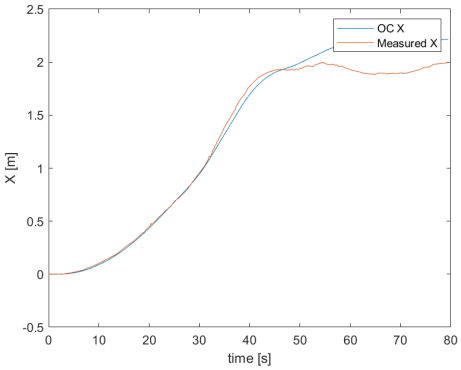
Figure 35: Test 2 time optimal solution, as computed using the GPOPS solver.



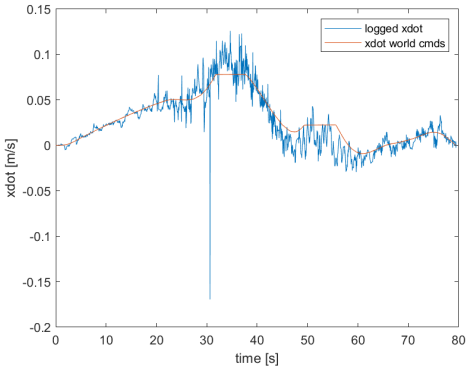
(a) Optimal Trajectory



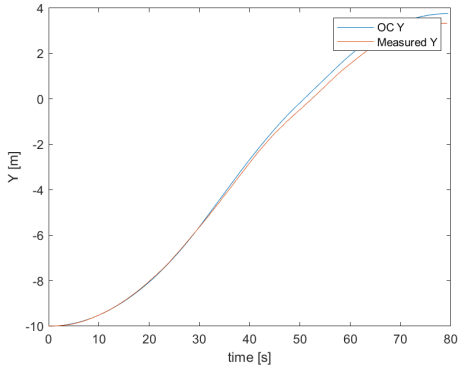
(b) Compared Trajectory



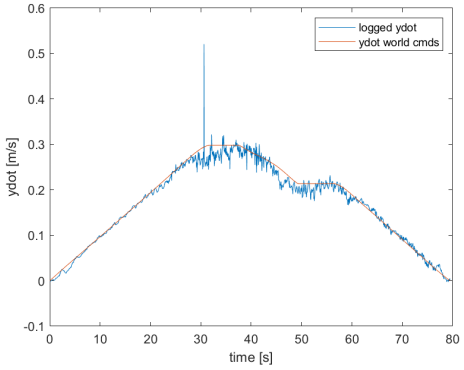
(c) Compared X



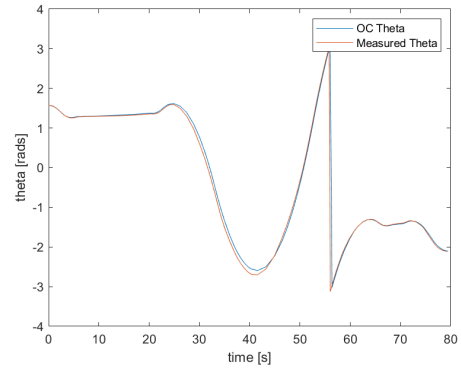
(d) Compared X Velocity



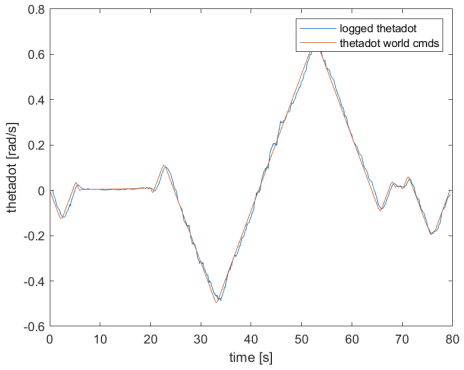
(e) Compared Y



(f) Compared Y Velocity



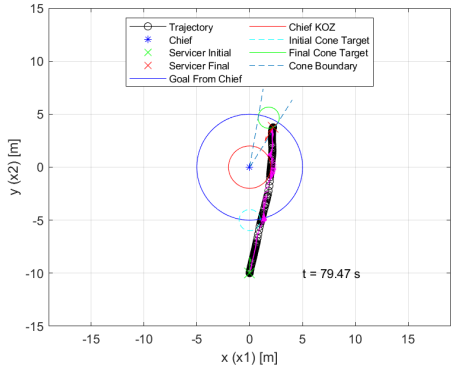
(g) Compared Theta



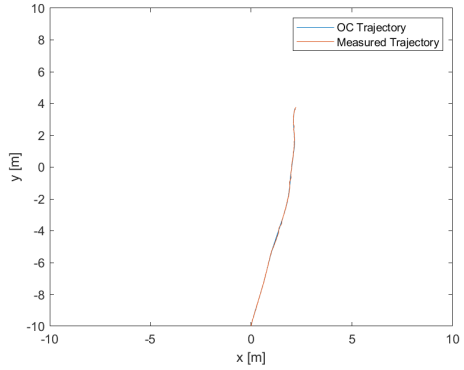
(h) Compared Theta Velocity

Figure 36: Test 2 hardware open loop control with data collected off the omnibot robot in the ANT motion capture lab, compared to the optimal trajectory shown in Figure 35

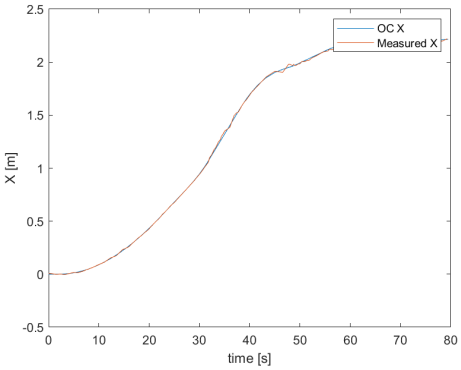
Test Case 2 is then demonstrated on the omnibot hardware using the closed loop controller from Algorithm 2 in Section 3.3.3.3. The results are shown in Figure 37 and when compared to Figure 36 highlight the impact of the closed loop controller in removing the accumulated error via the feedback signal shown in Figure 38. As further shown in Figures 37c, 37e and 37g, the pose of the omnibot closely tracks the optimal control trajectory with an RMSE of 9.6 mm in x, 13.7 mm in y, and 0.0502 rad in θ. While the omnibot pose tracks the trajectory closer with the closed loop controller, the state velocities shown in Figures 37d, 37f and 37h are noticeably noisier. This is primarily due to the feedback signal in Figure 38 being added to the commanded velocities. The one tenth scaling increases the difficulty for the feedback controller as the motion capture system is only able to resolve robot position on the order of millimeters and the scaled x velocities in this test case are only a few mm/s.



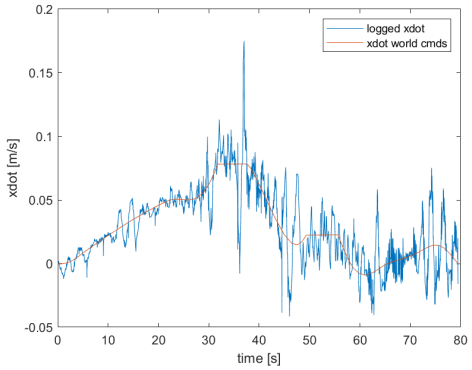
(a) Optimal Trajectory



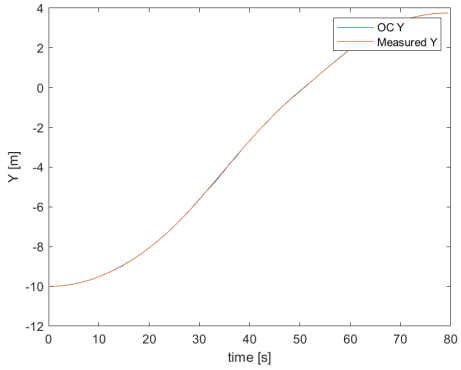
(b) Compared Trajectory



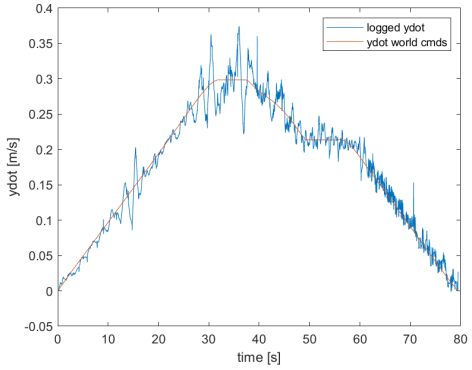
(c) Compared X



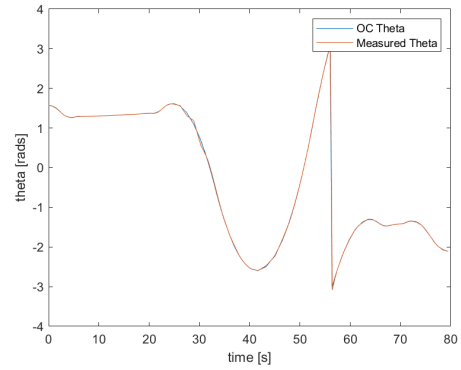
(d) Compared X Velocity



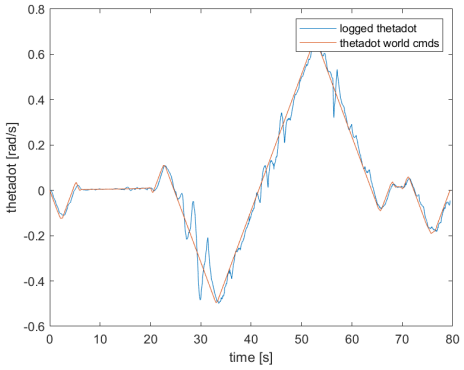
(e) Compared Y



(f) Compared Y Velocity



(g) Compared Theta



(h) Compared Theta Velocity

Figure 37: Test 2 hardware closed loop control with data collected off the omnibot robot in the ANT motion capture lab, compared to the optimal trajectory shown in Figure 35

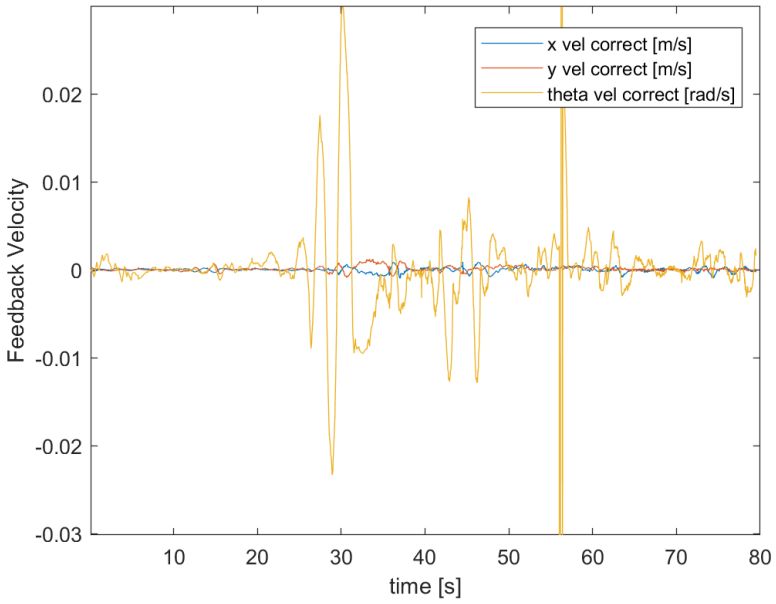


Figure 38: Test 2 feedback signal applied during the hardware closed loop control test to correct for error in the omnibot pose measured by the VICON system. A spike is shown at around 56 seconds due to an issue in the feedback controller when the orientation wraps from π to $-\pi$.

4.2.3 Test Case 3: Rotating Chief 2

The third test case uses the same initial conditions and constant chief rotation rate of two degrees per second as Test Case 2 except the chief initial orientation is zero rad instead of $-\pi/2$ rad. The resulting trajectory is shown in Figure 40 and is smoother than Test Case 2 due to the lack of interaction with the keep out zone constraint. The control history in Figure 40b offers a clear example of bang-off-bang control as mentioned in Section 4.2.1.

The closed loop controller is then used to demonstrate the optimal trajectory via hardware in Figure 41 and simulation in Figure 42. The feedback signals applied throughout each demonstration are shown in Figure 43. While the hardware theta correction is centered around zero in Figure 43a, the simulated theta correction has a

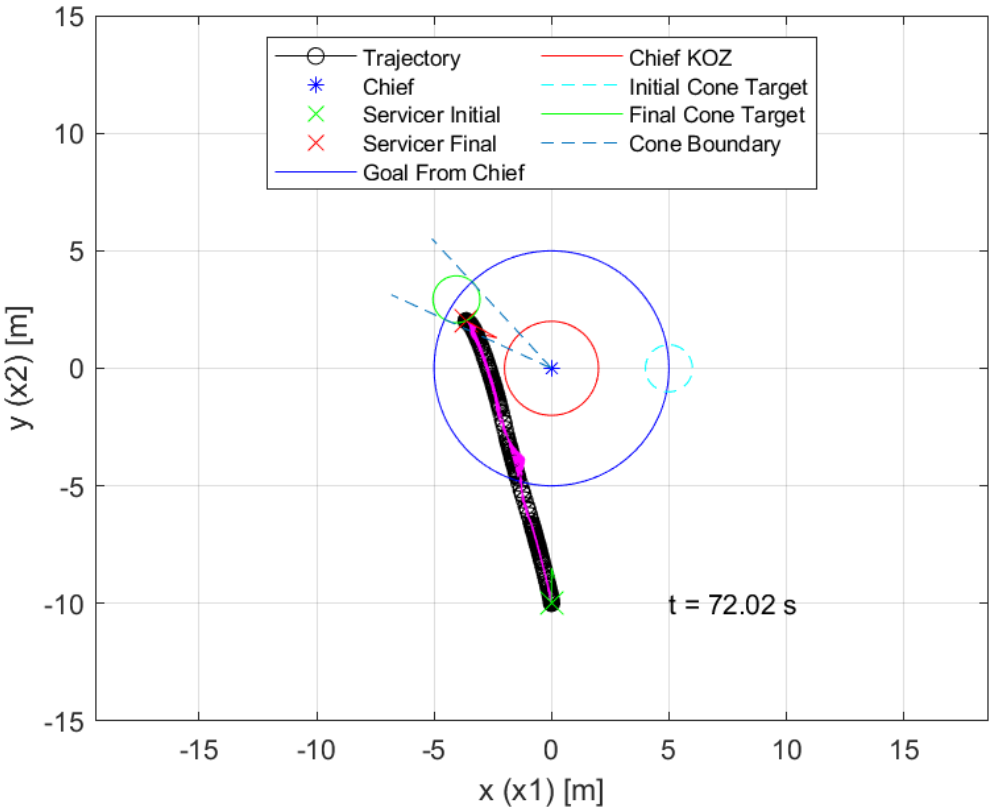
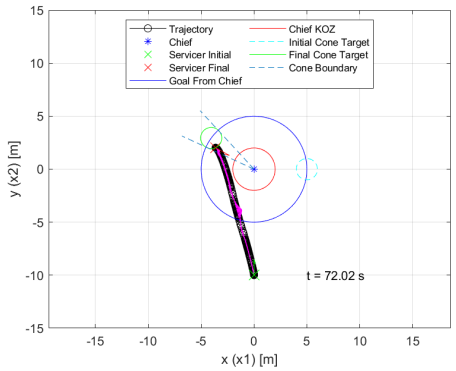
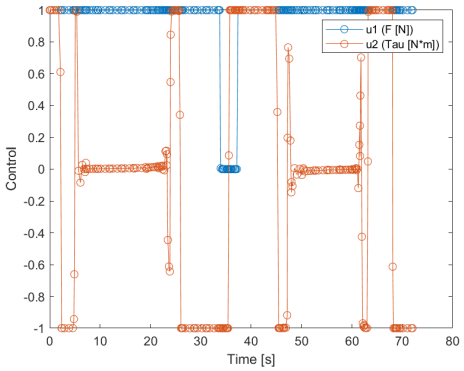


Figure 39: Test Case 3 Optimal Solution: Trajectory

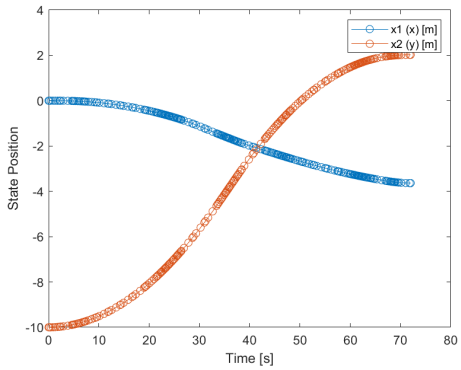
negative bias that can be seen in that is correcting for the orientation issue mentioned in Section 4.2.1. Due to being one of the smoothest trajectories, this test yields some of the lowest RMSE values using the hardware at 7.2 mm in x, 7.5 mm in y, and 0.0039 rad in θ. This compares to the results from the simulation with RMSE values of 1.5 mm in x, 2.0 mm in y, and 0.0205 rad in θ.



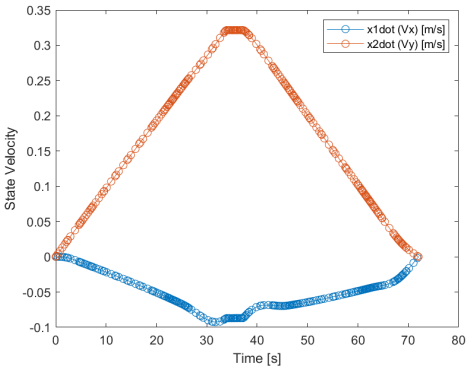
(a) Trajectory



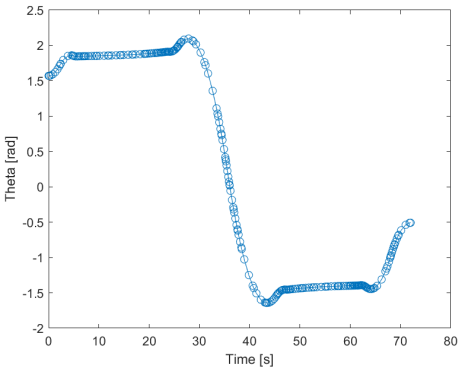
(b) Control History



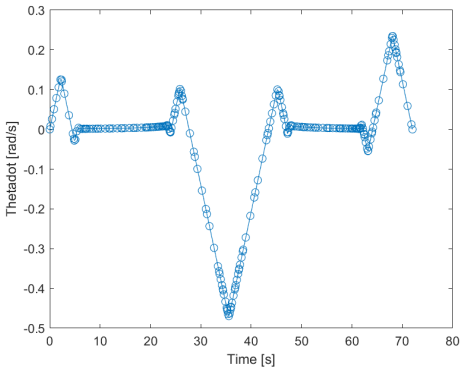
(c) Position



(d) Planar Velocities

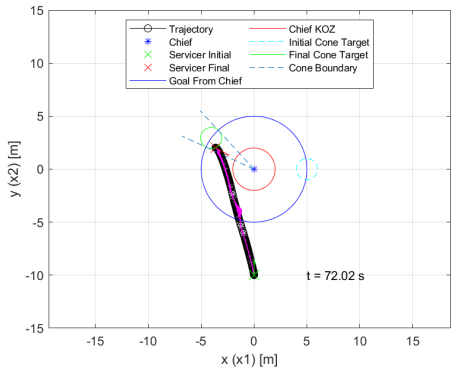


(e) Orientation

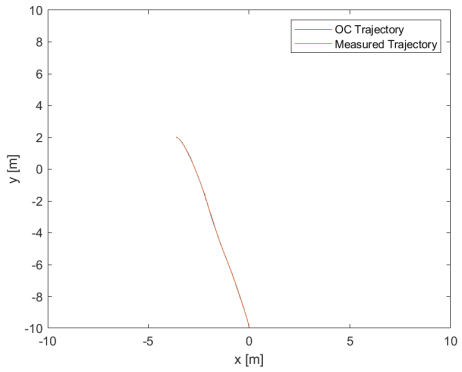


(f) Angular Velocity

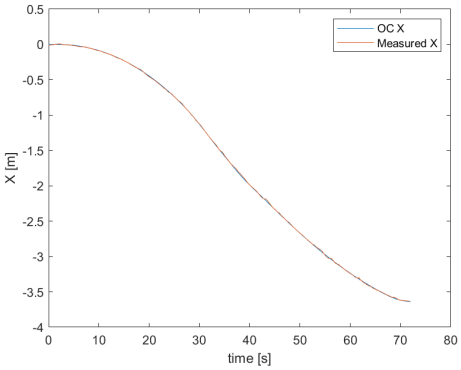
Figure 40: Test 3 time optimal solution, as computed using the GPOPS solver.



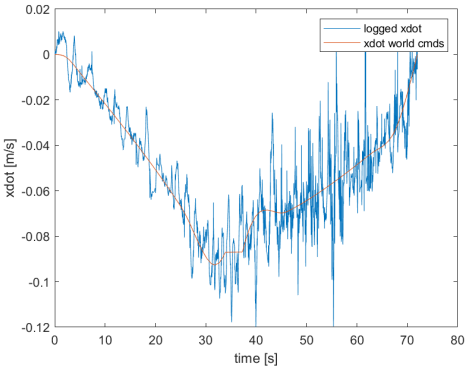
(a) Optimal Trajectory



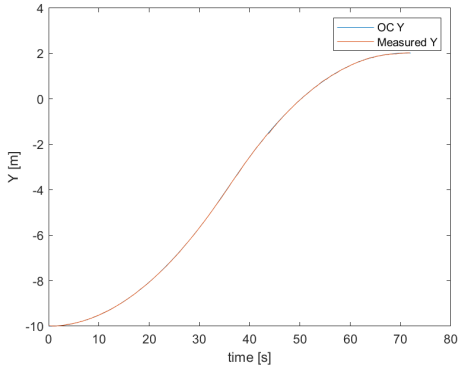
(b) Compared Trajectory



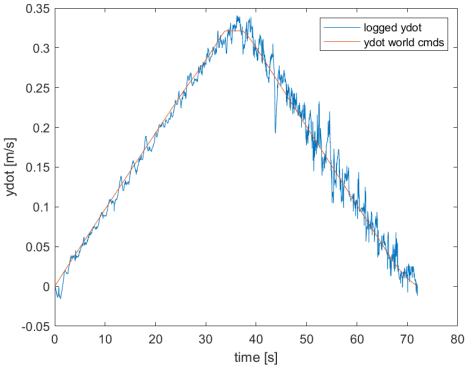
(c) Compared X



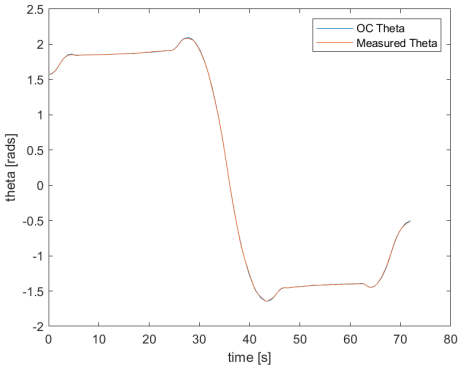
(d) Compared X Velocity



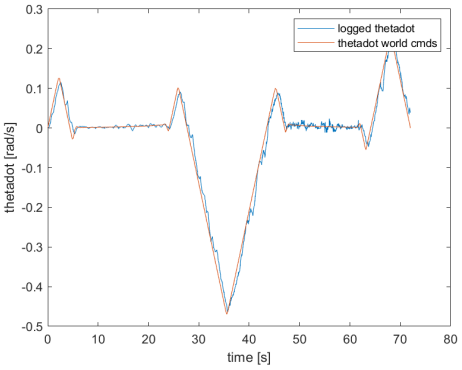
(e) Compared Y



(f) Compared Y Velocity

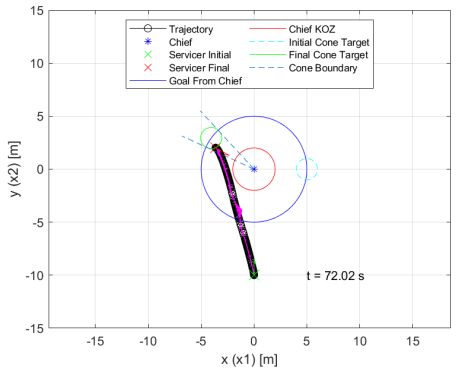


(g) Compared Theta

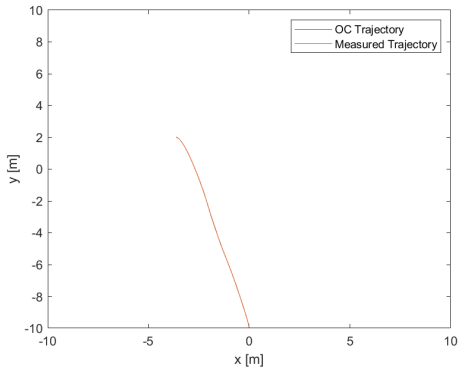


(h) Compared Theta Velocity

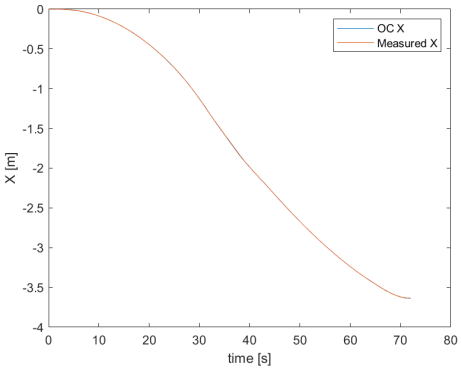
Figure 41: Test 3 hardware closed loop control with data collected off the omnibot robot in the ANT motion capture lab, compared to the optimal trajectory shown in Figure 40



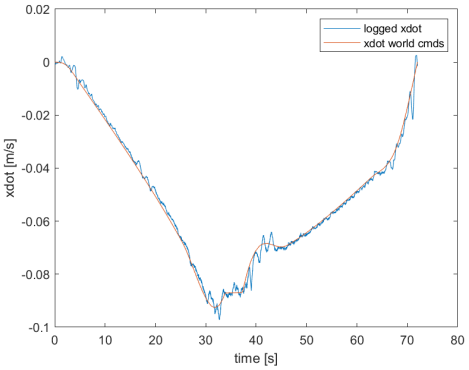
(a) Optimal Trajectory



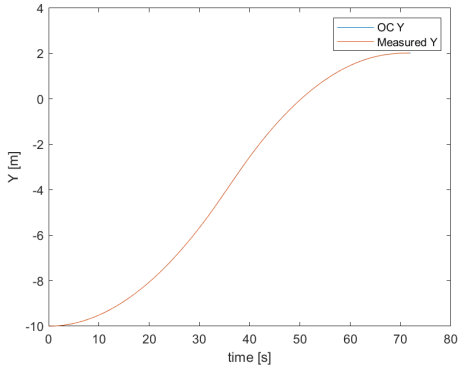
(b) Compared Trajectory



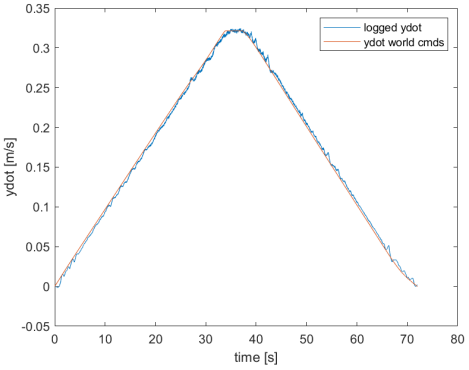
(c) Compared X



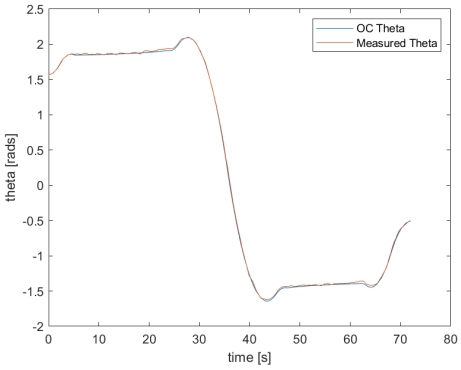
(d) Compared X Velocity



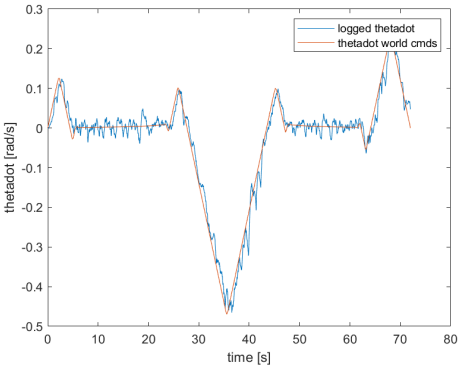
(e) Compared Y



(f) Compared Y Velocity

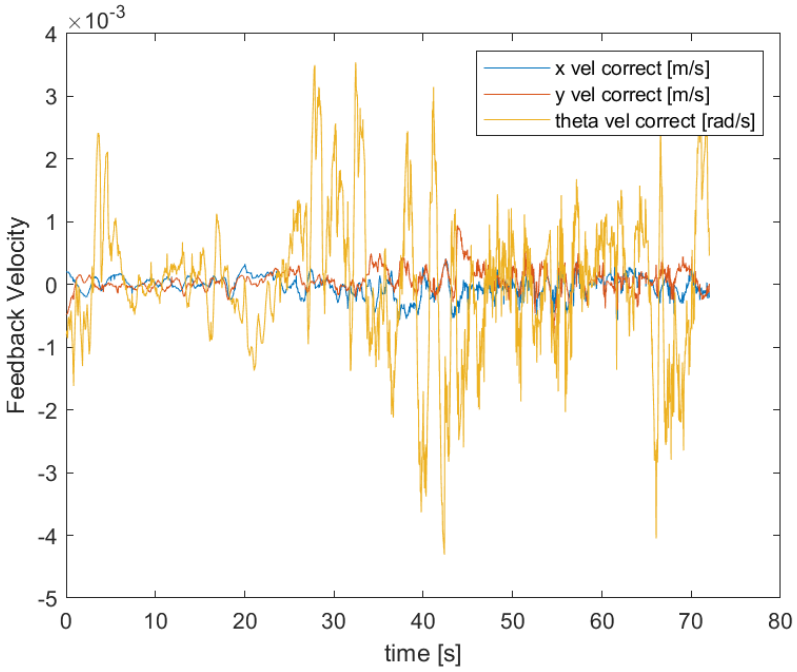


(g) Compared Theta

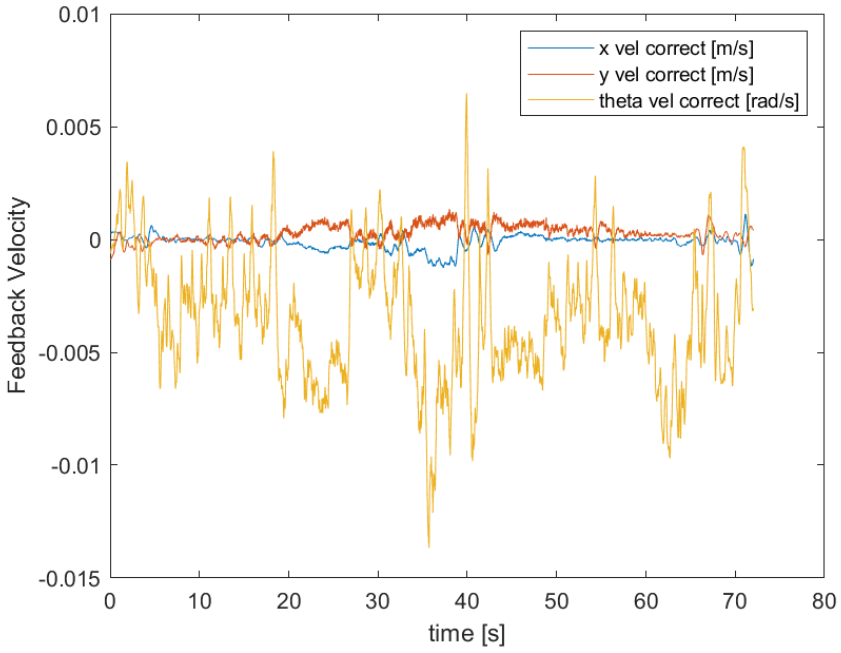


(h) Compared Theta Velocity

Figure 42: Test 3 simulated closed loop control compared to the optimal control trajectory in Figure 40



(a) Test 3 Hardware Test Feedback Signal



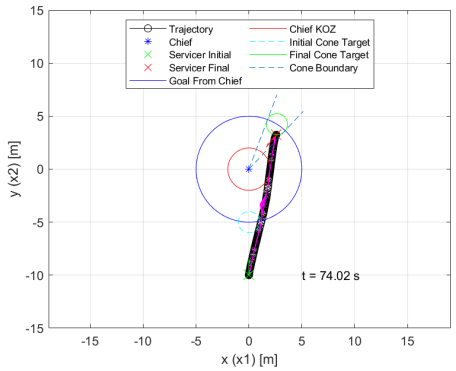
(b) Test 3 Simulated Test Feedback Signal

Figure 43: Compared velocity feedback signals applied to the hardware and simulated demonstrations in Test Case 3. A negative bias is visible in the theta feedback signal in Figure 43b correcting for an issue with orientation in the simulator.

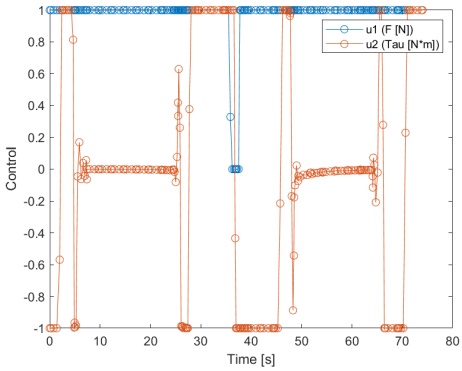
4.2.4 Test Case 4: Approximated Dynamics 1

While the first three test cases use the CW equations of motion for state dynamics, the following two test cases use the double integrator (DI) based free-flying model introduced in Section 2.1.2.3. Test Case 4 uses the same initial conditions as Test Case 2 but with the DI dynamics. The main difference in the result as shown in Figure 44 is the DI dynamics remove the interaction with the keep out zone constraint as present in Figure 35.

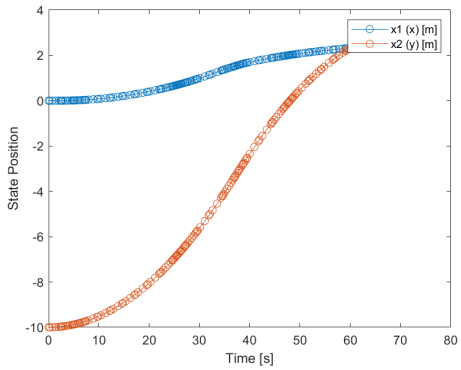
Figure 45 highlights the differences between the trajectories in Test Cases 2 and 4. Since the deputy does not need to counteract the additional terms in the CW dynamics to maneuver around the keep out zone, it is able to apply more thrust towards reaching the objective and reaches the terminal conditions approximately 5 seconds faster. However, this trajectory demonstrates a scenario where the double integrator approximation would not be sufficient in an actual orbital scenario, and highlights the importance of the testbed being able to utilize the CW equations. Regardless, the testbed is shown to be capable of demonstrating this trajectory via hardware in Figure 46.



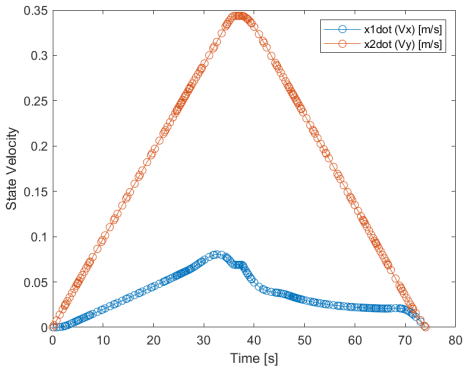
(a) Trajectory



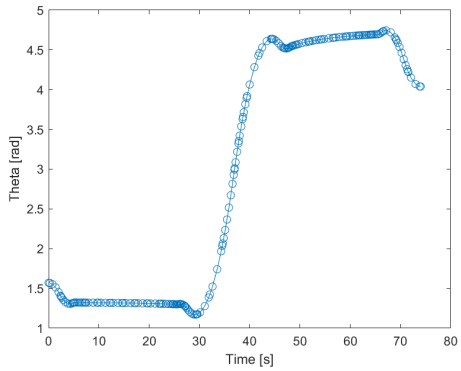
(b) Control History



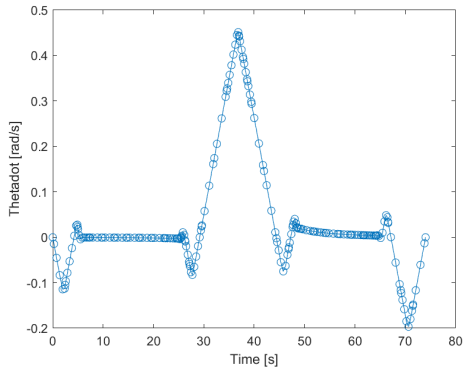
(c) Position



(d) Planar Velocities

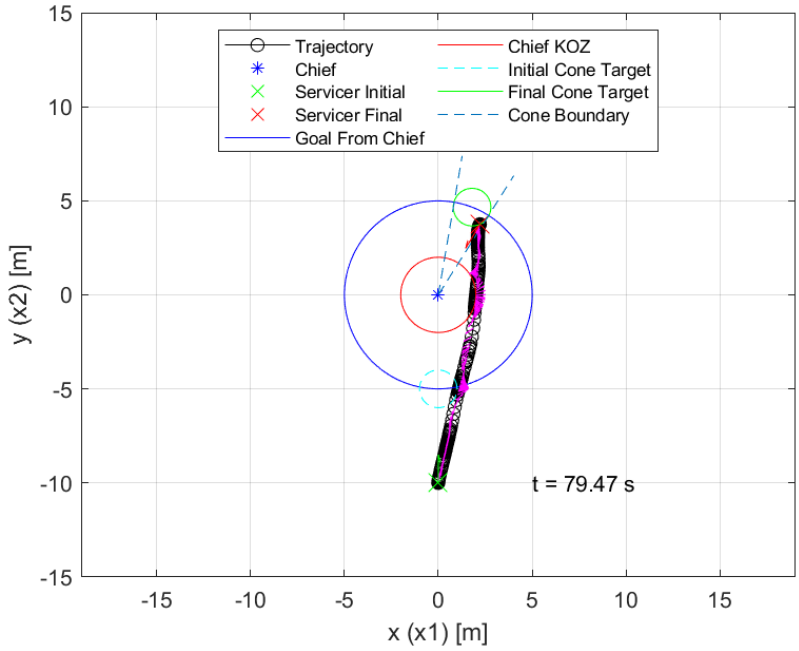


(e) Orientation

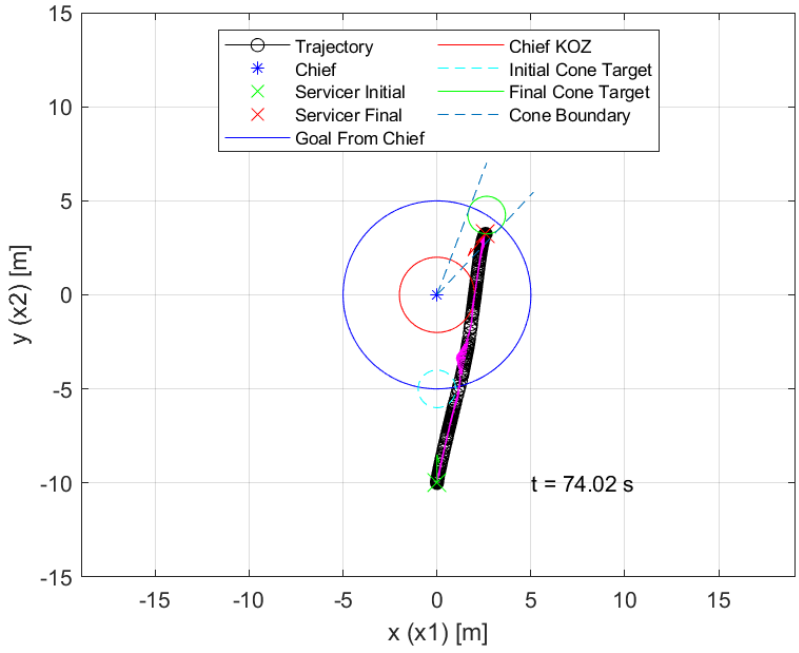


(f) Angular Velocity

Figure 44: Test 4 time optimal solution, as computed using the GPOPS solver.

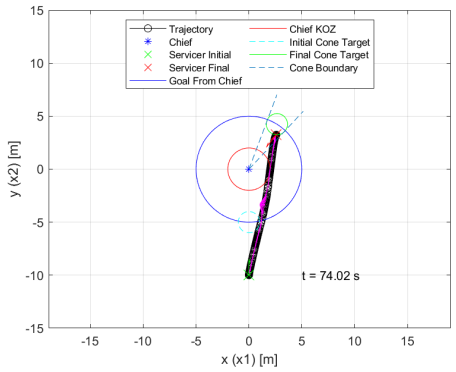


(a) Test 2 Clohessy-Wiltshire

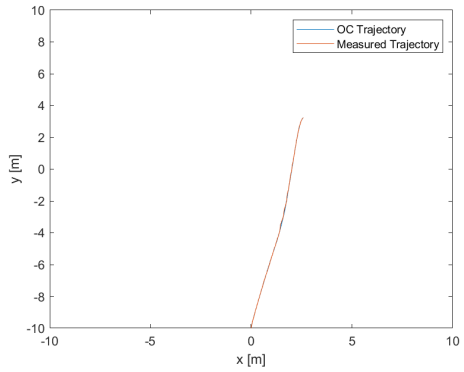


(b) Test 4 Double Integrator

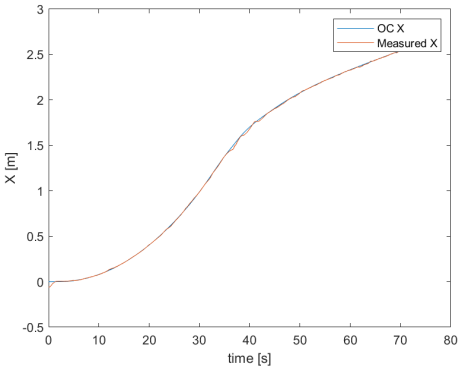
Figure 45: Compared dynamics of Test 2 using CW dynamics vs Test 4 using DI dynamics



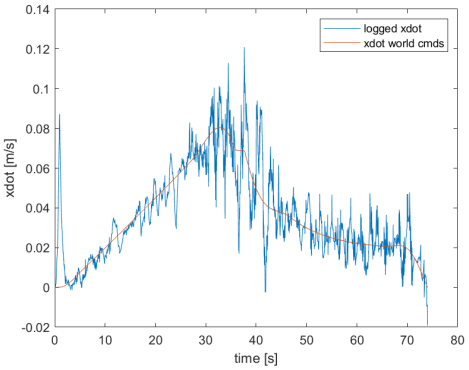
(a) Optimal Trajectory



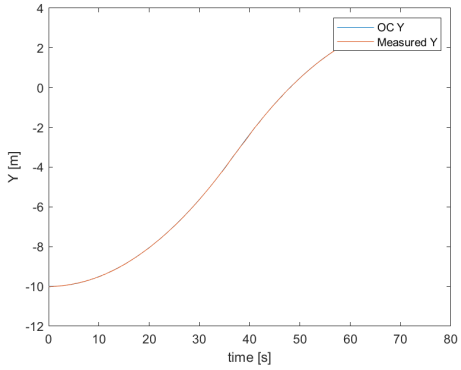
(b) Compared Trajectory



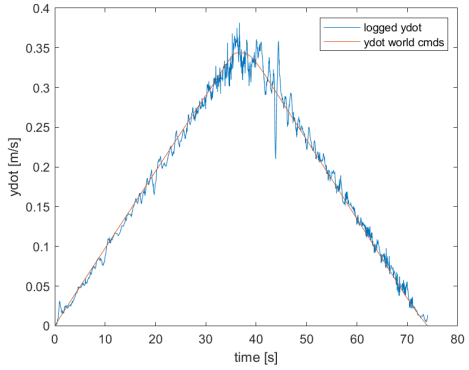
(c) Compared X



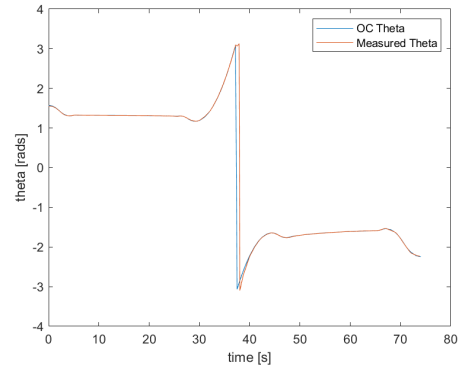
(d) Compared X Velocity



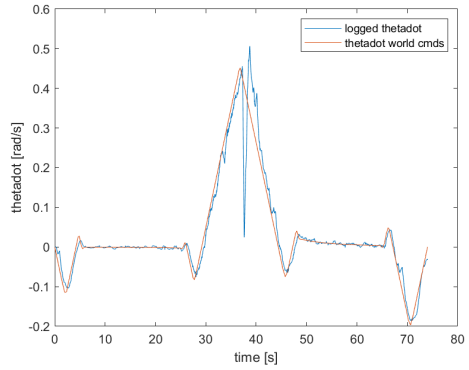
(e) Compared Y



(f) Compared Y Velocity



(g) Compared Theta

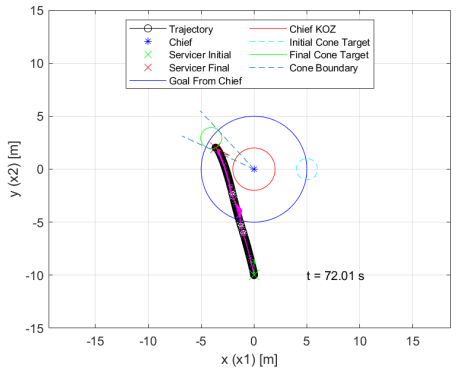


(h) Compared Theta Velocity

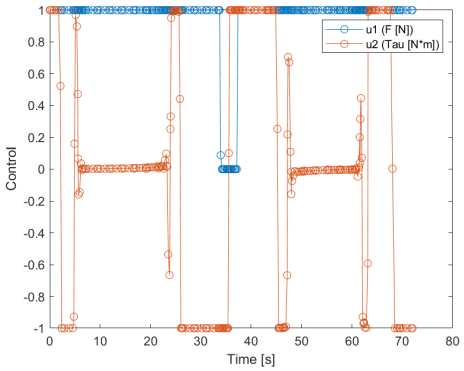
Figure 46: Test 4 hardware closed loop control with data collected off the omnibot robot in the ANT motion capture lab, compared to the optimal trajectory shown in Figure 44

4.2.5 Test Case 5: Approximated Dynamics 2

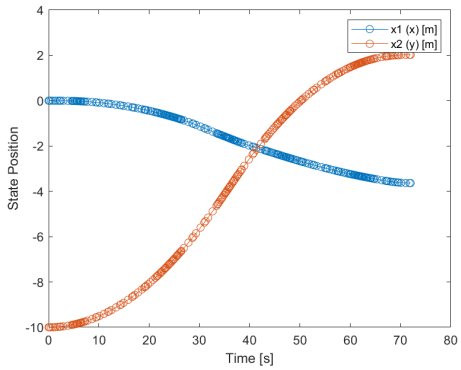
Similar to Test Case 4, Test Case 5 approximates Test Case 3 with DI dynamics instead of using CW dynamics. However, unlike Test Case 4, this test case is successful in approximating the optimal trajectory with DI dynamics as shown in Figure 47. Figure 48 shows the difference in the using the two state dynamics. The duration of the two trajectories is only different by 0.01 s. This is primarily due to no significant interaction with the keep out zone that would require additional maneuvering under the CW dynamics. The DI approximated trajectory is then demonstrated on the omnibot hardware as shown in Figure 49.



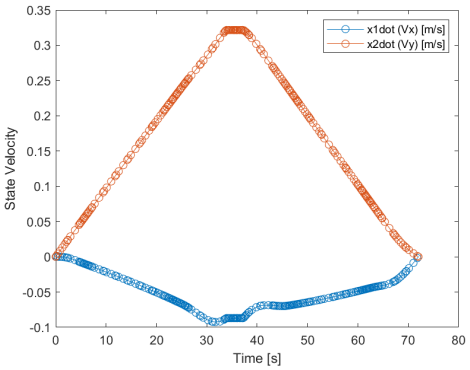
(a) Trajectory



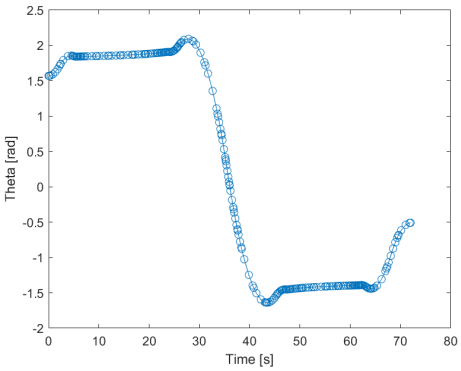
(b) Control History



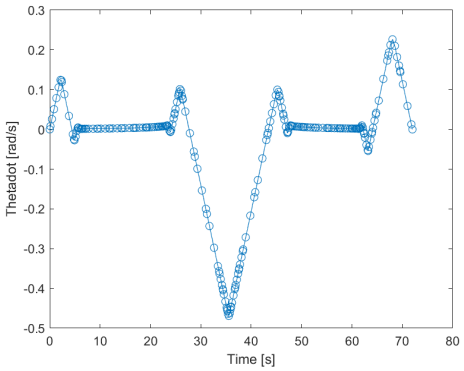
(c) Position



(d) Planar Velocities

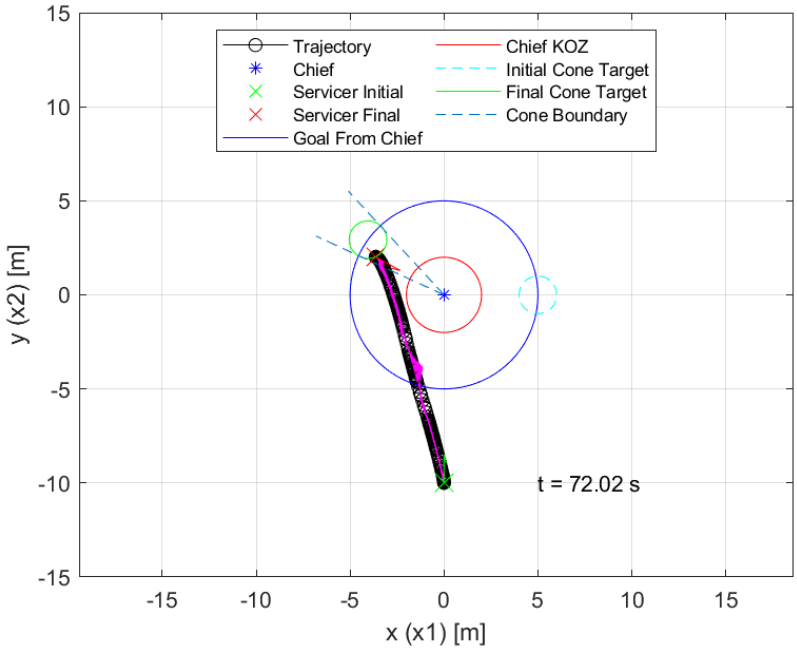


(e) Orientation

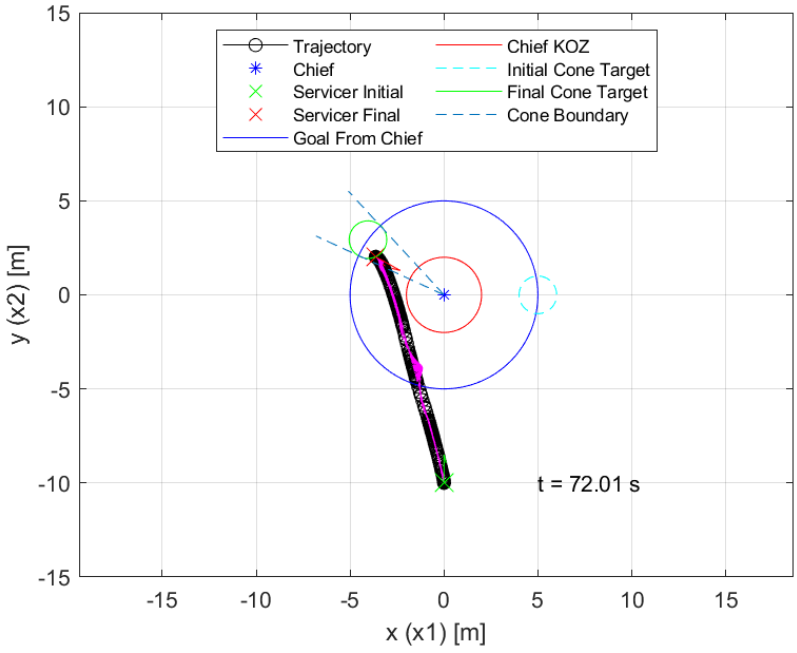


(f) Angular Velocity

Figure 47: Test 5 time optimal solution, as computed using the GPOPS solver.

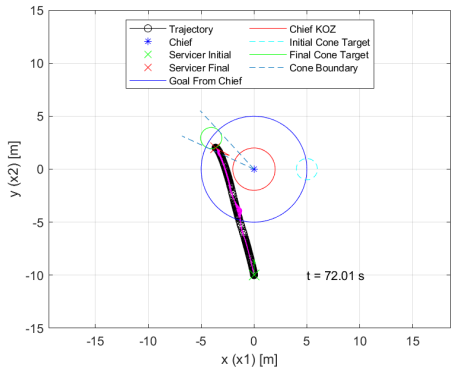


(a) Test 3 Clohessy-Wiltshire

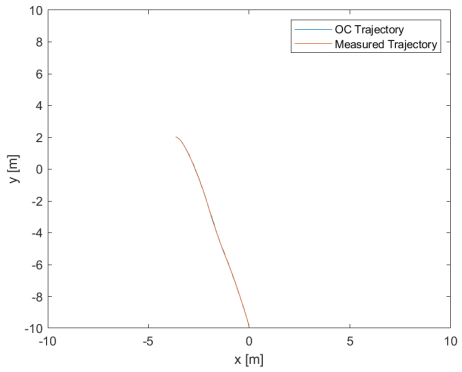


(b) Test 5 Double Integrator

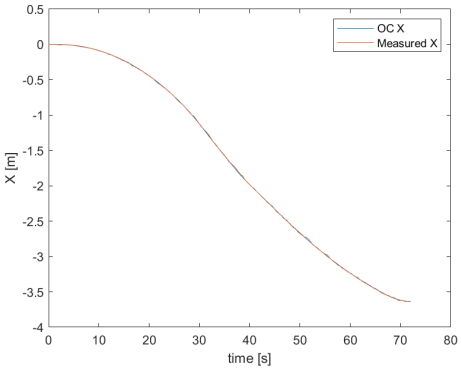
Figure 48: Compared dynamics of Test 3 using CW dynamics vs Test 5 using DI dynamics.



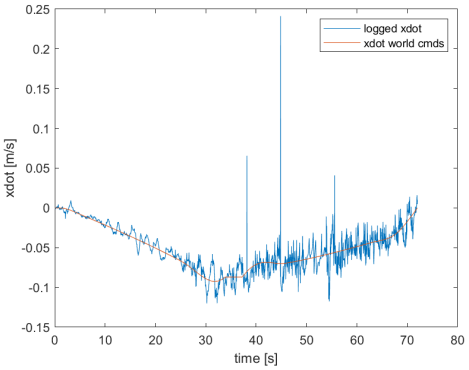
(a) Optimal Trajectory



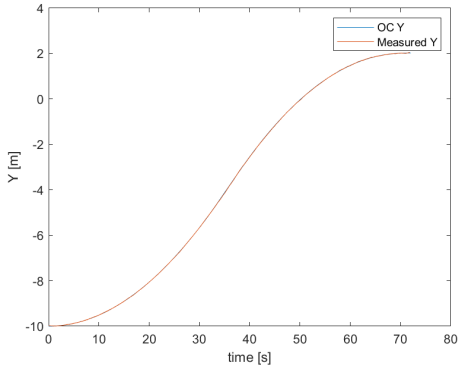
(b) Compared Trajectory



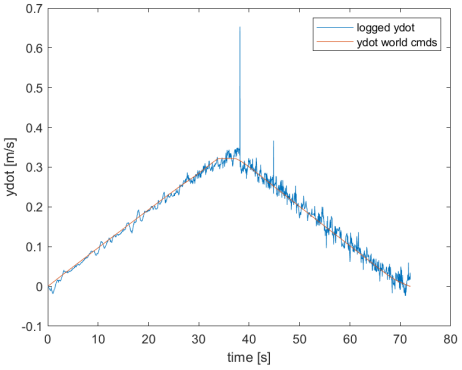
(c) Compared X



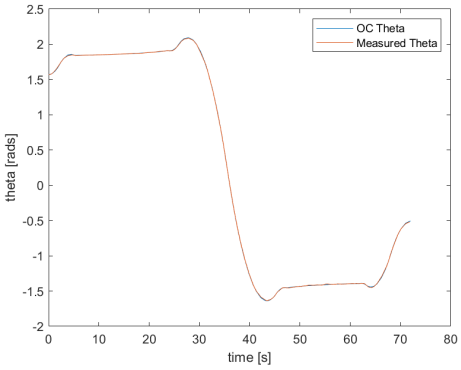
(d) Compared X Velocity



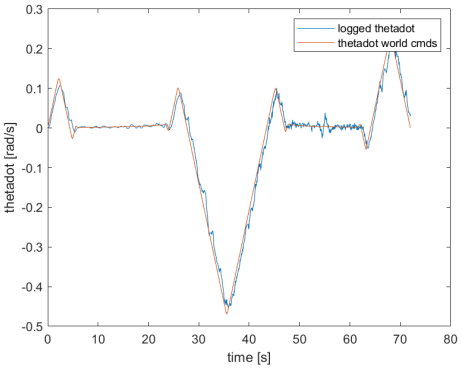
(e) Compared Y



(f) Compared Y Velocity



(g) Compared Theta



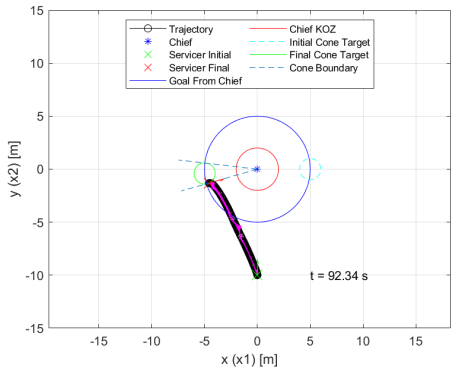
(h) Compared Theta Velocity

Figure 49: Test 5 hardware closed loop control with data collected off the omnibot robot in the ANT motion capture lab, compared to the optimal trajectory shown in Figure 47

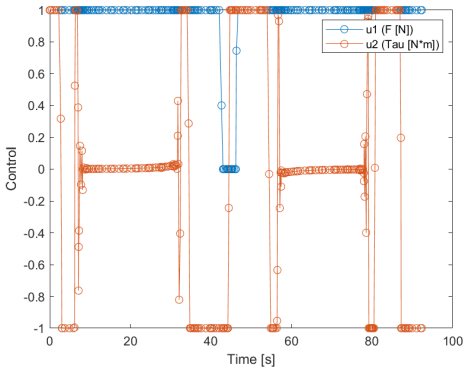
4.2.6 Test Case 6: Double Mass Deputy

The final two test cases demonstrate the flexibility of the hardware testbed to demonstrate deputy satellites with different physical configurations. Any initial condition can be modified in the optimal control formulation, including maximum force and torque limits, mass, moment of inertia, and the chief rotation rate. Test Case 6 uses all the same initial conditions and the CW dynamics as Test Case 3, but the deputy is modeled with twice as much mass (200 kg vs 100 kg). The time optimal control solution is found in Figure 50.

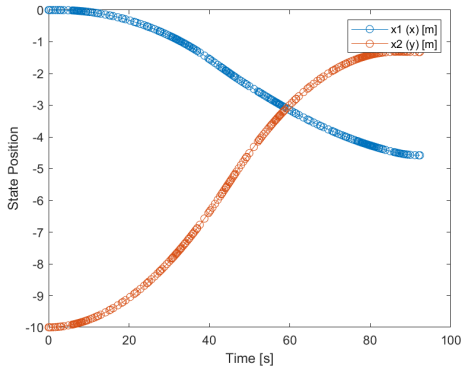
The optimal trajectories for Test Case 3 and 6 are compared in Figure 51. As expected, the deputy in this scenario with twice as much mass but the same thrust constraints takes approximately 20 seconds longer to reach the terminal conditions. The effect of the mass can also be seen in the planar velocities in Figure 50d, where it peaks below 0.2 m/s vs 0.33 m/s in Test Case 3. This trajectory is then demonstrated on the omnibot hardware as shown in Figure 52.



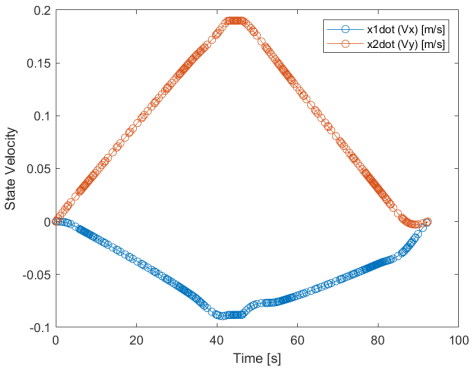
(a) Trajectory



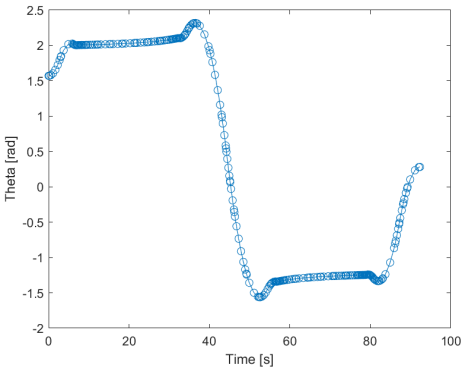
(b) Control History



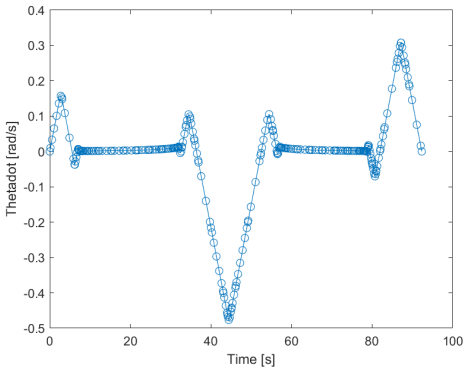
(c) Position



(d) Planar Velocities

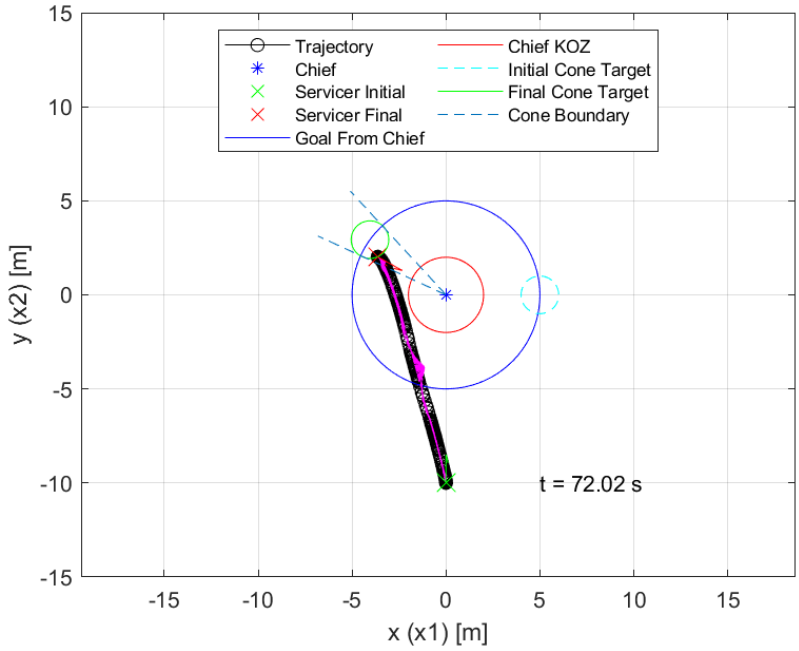


(e) Orientation

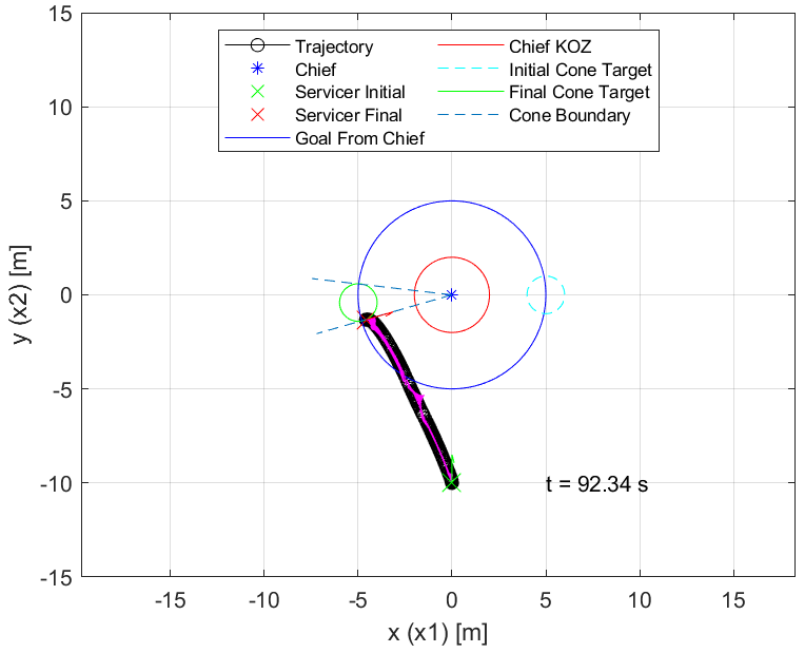


(f) Angular Velocity

Figure 50: Test 6 time optimal solution, as computed using the GPOPS solver.

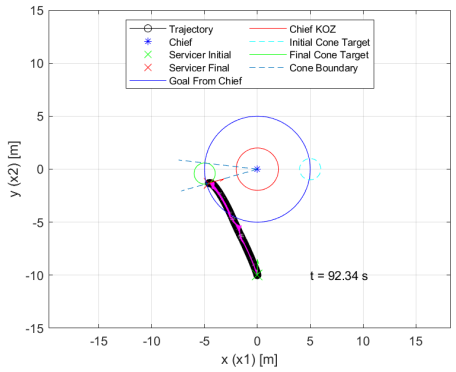


(a) Test 3 100 kg Deputy

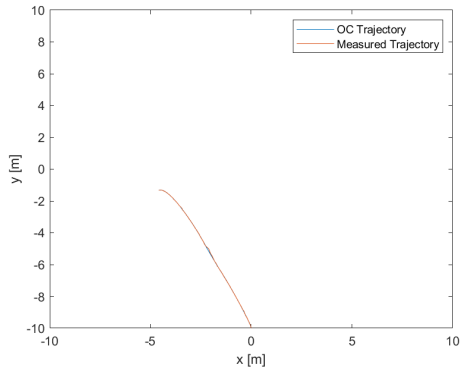


(b) Test 6 200 kg Deputy

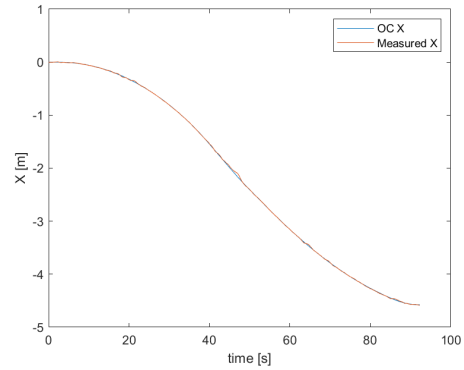
Figure 51: Compared mass of Test 3 using a 100 kg deputy vs Test 6 using a 200 kg deputy.



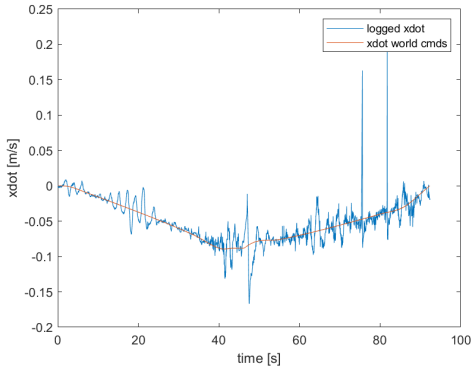
(a) Optimal Trajectory



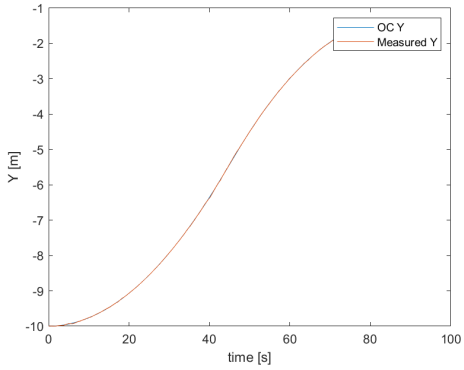
(b) Compared Trajectory



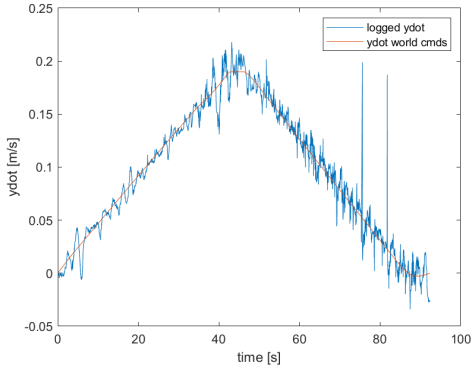
(c) Compared X



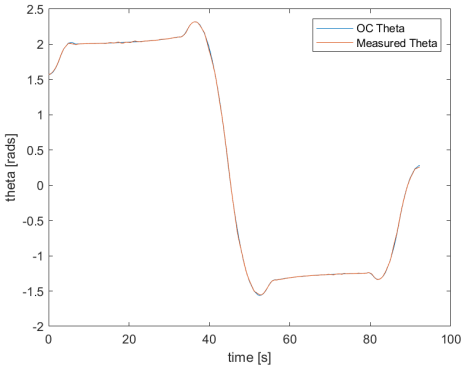
(d) Compared X Velocity



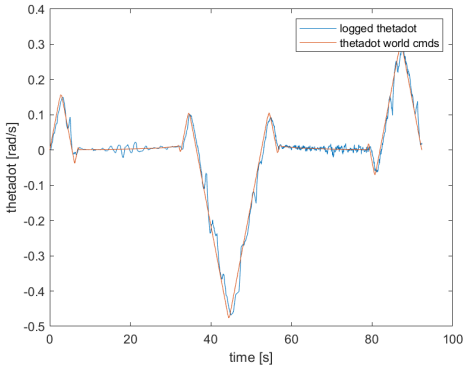
(e) Compared Y



(f) Compared Y Velocity



(g) Compared Theta



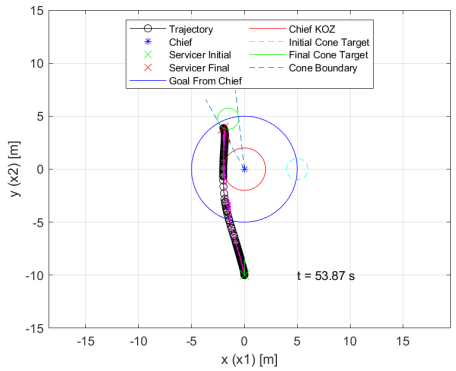
(h) Compared Theta Velocity

Figure 52: Test 6 hardware closed loop control with data collected off the omnibot robot in the ANT motion capture lab, compared to the optimal trajectory shown in Figure 50

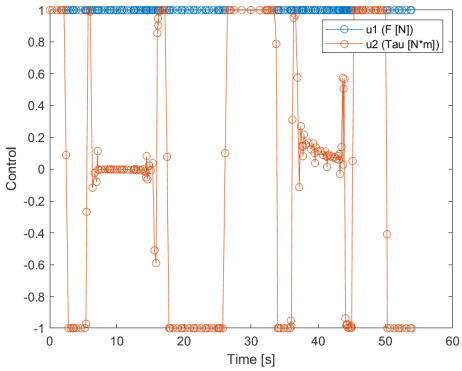
4.2.7 Test Case 7: Half Mass Deputy

Similar to Test Case 6, Test Case 7 uses the same setup as Test Case 3, but the deputy is modeled with half as much mass (50 kg vs 100 kg). The time optimal control solution is found in Figure 53.

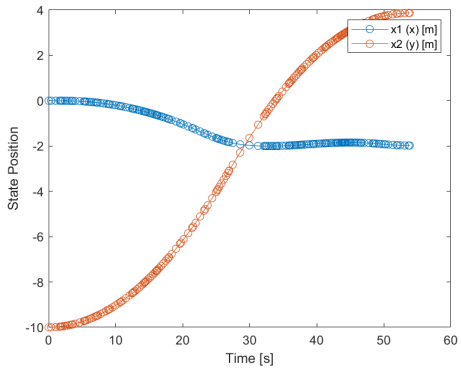
The optimal trajectories for Test Cases 3 and 7 are compared in Figure 54. As expected, the deputy in this scenario with half as much mass but the same thrust constraints reaches the terminal conditions approximately 18 seconds faster. This trajectory is then demonstrated on the omnibot hardware as shown in Figure 55.



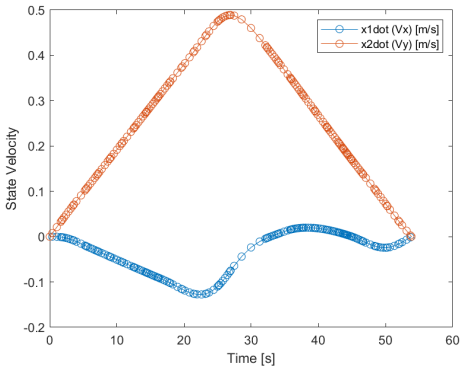
(a) Trajectory



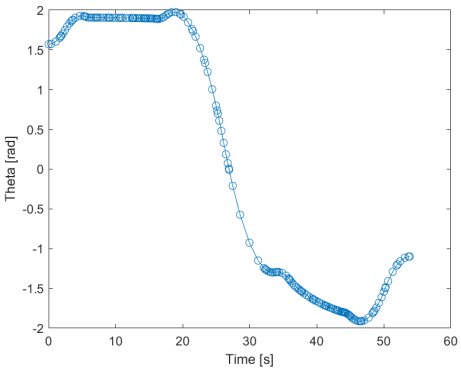
(b) Control History



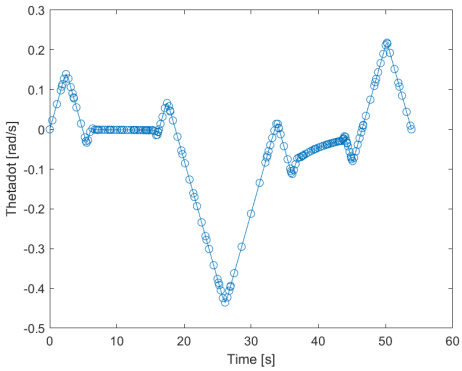
(c) Position



(d) Planar Velocities

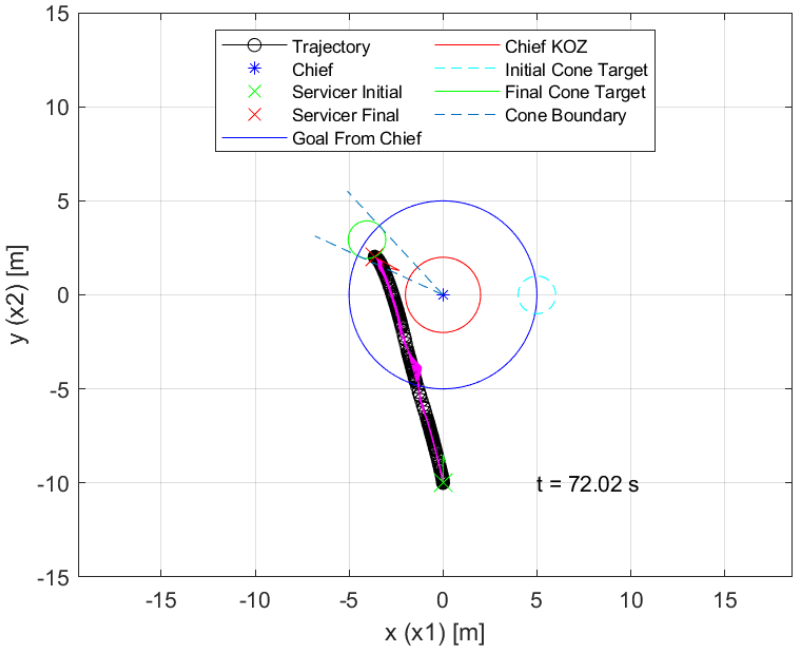


(e) Orientation

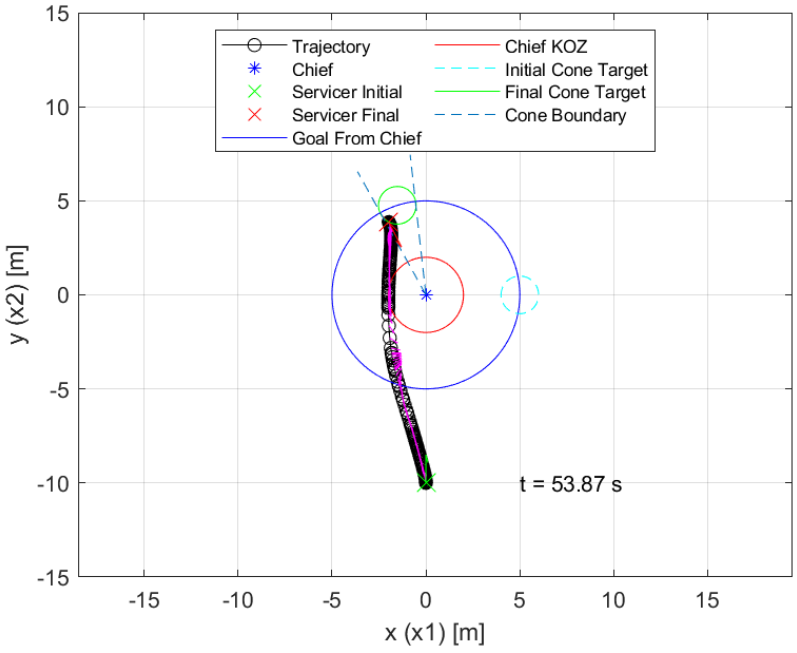


(f) Angular Velocity

Figure 53: Test 7 time optimal solution, as computed using the GPOPS solver.

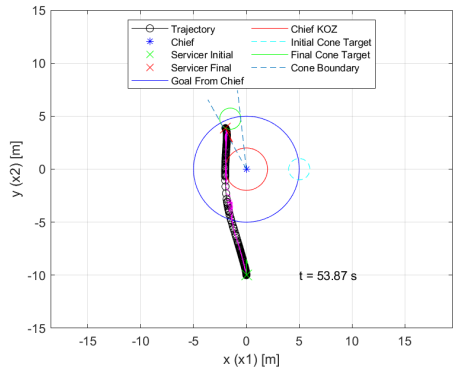


(a) Test 3 100 kg Deputy

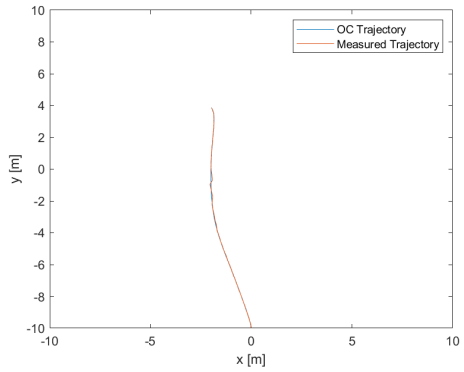


(b) Test 7 50 kg Deputy

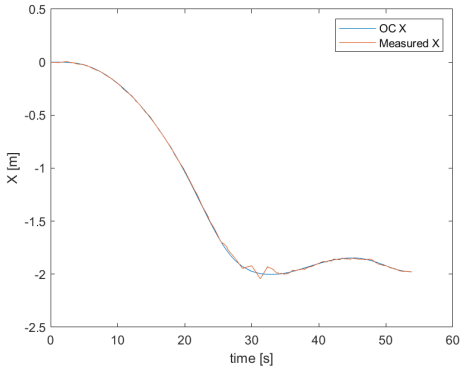
Figure 54: Compared mass of Test 3 using a 100 kg deputy vs Test 7 using a 50 kg deputy.



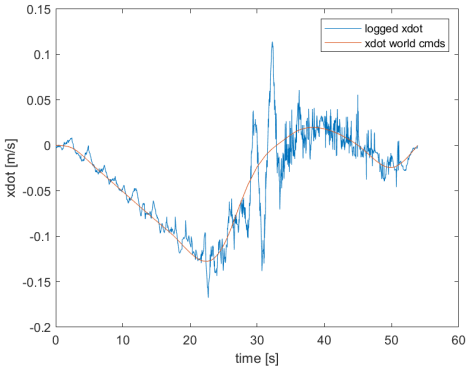
(a) Optimal Trajectory



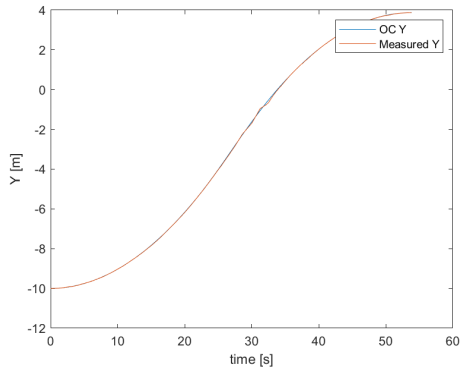
(b) Compared Trajectory



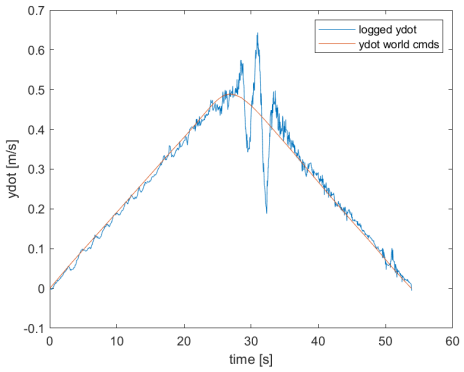
(c) Compared X



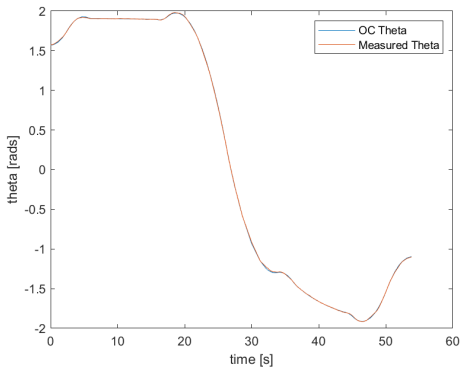
(d) Compared X Velocity



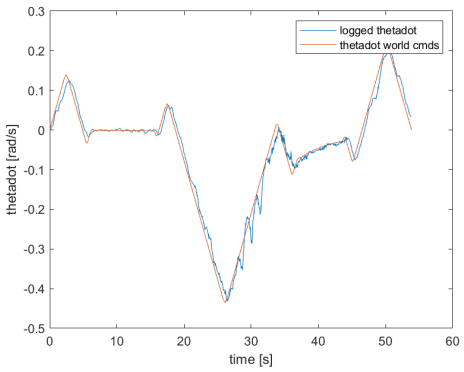
(e) Compared Y



(f) Compared Y Velocity



(g) Compared Theta



(h) Compared Theta Velocity

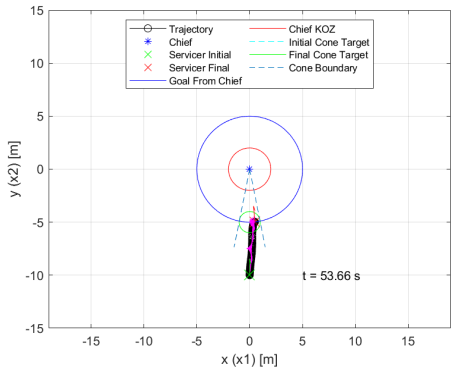
Figure 55: Test 7 hardware closed loop control with data collected off the omnibot robot in the ANT motion capture lab, compared to the optimal trajectory shown in Figure 53

4.3 Full Scale Testing

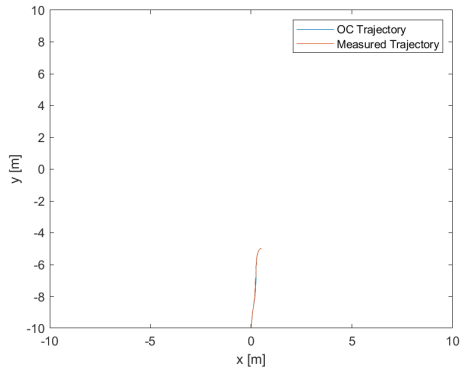
While all of the tests in Section 4.2 are performed at 10% scaling to fit within the usable area of the ANT VICON chamber, the following demonstrations utilize a much larger test area outfitted with 60 VICON cameras. The usable area in these tests is 8x8 m vs the 1x1 m area used in Section 4.2. Therefore, scaling still exists, but with 80% scaling the measurements errors are only multiplied by 1.25x instead of 10x when scaling back up to compare the resulting trajectory to the optimal control solution. The first three test cases from Section 4.2 are demonstrated at 80% scaling in Figures 56 to 58.

However, while the omnibot hardware is capable of performing Test Case 1 at 80% scaling, the other two test cases suffer significant tracking errors during peak velocities. This is due to to the maximum hardware velocity of 0.22 m/s. A comparison of the maximum hardware velocity and the peak velocities required for each scenario is shown in Figures 59 to 61.

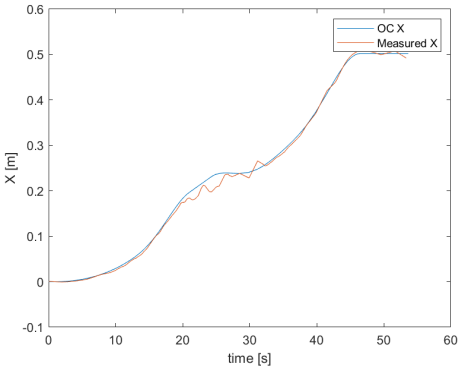
In order to successfully demonstrate the other two test cases, a scaling factor of 70% for Test Case 2 and 60% for Test Case 3 is applied to bring the peak velocities under the maximum hardware velocity limit. The results of these tests are shown in Figures 62 and 63.



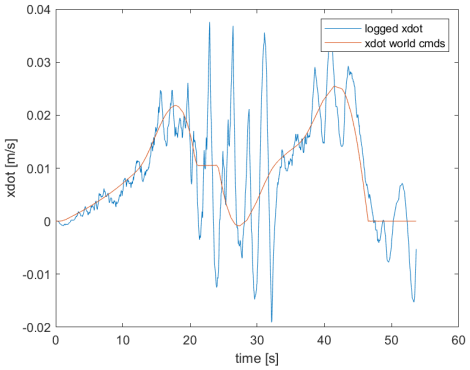
(a) Optimal Trajectory



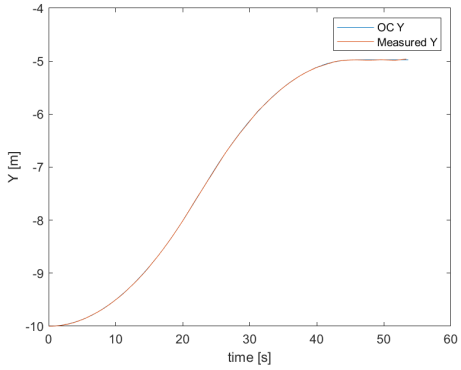
(b) Compared Trajectory



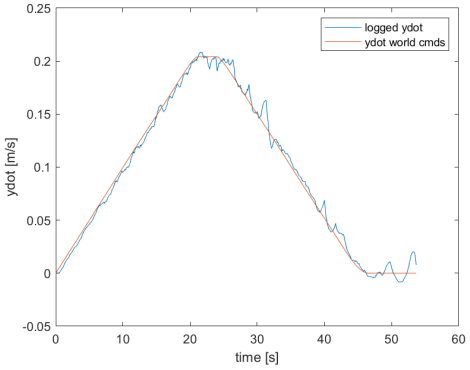
(c) Compared X



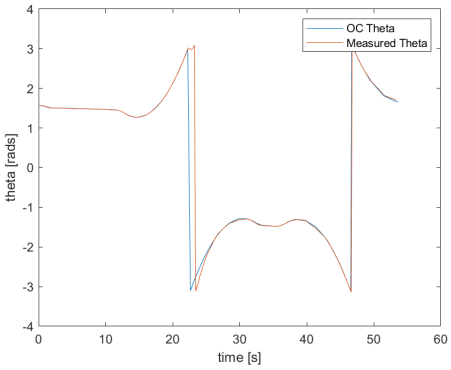
(d) Compared X Velocity



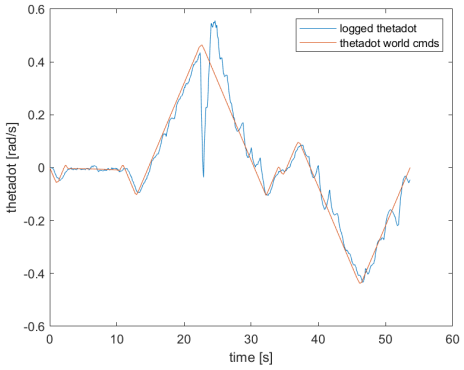
(e) Compared Y



(f) Compared Y Velocity

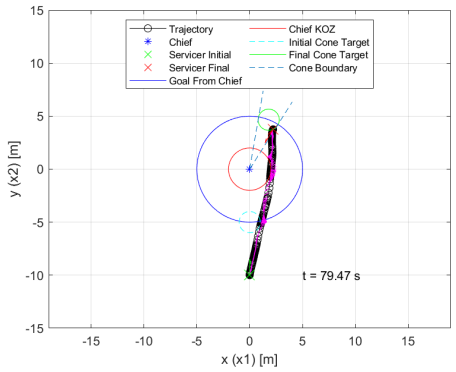


(g) Compared Theta

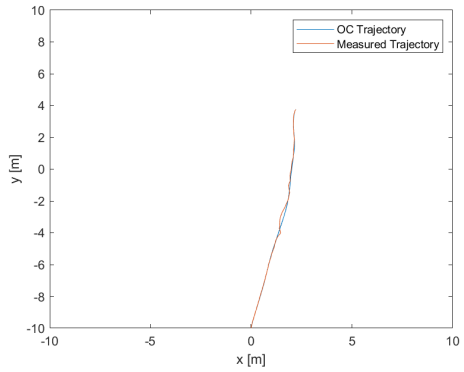


(h) Compared Theta Velocity

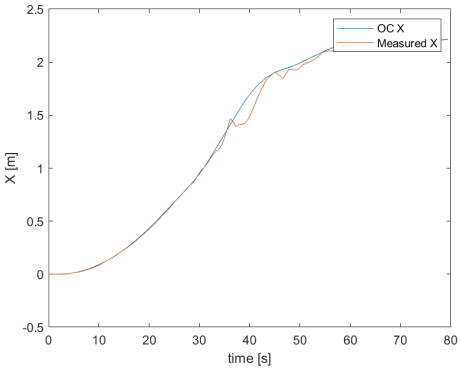
Figure 56: Test 1 MAV Lab 80% scaling, performed under maximum hardware velocity.



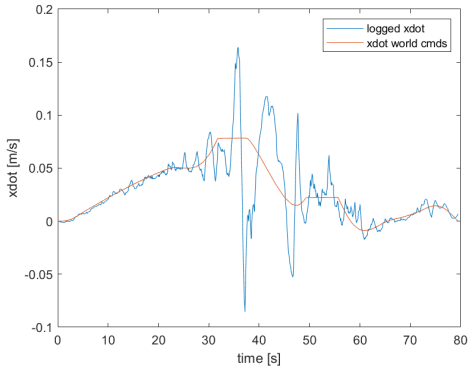
(a) Optimal Trajectory



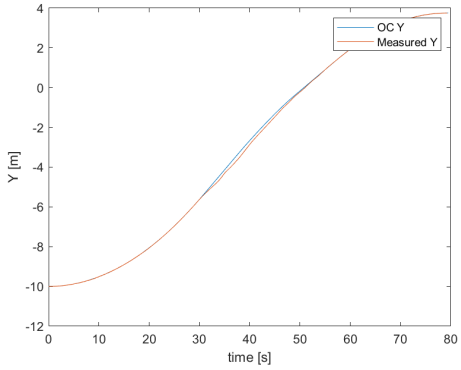
(b) Compared Trajectory



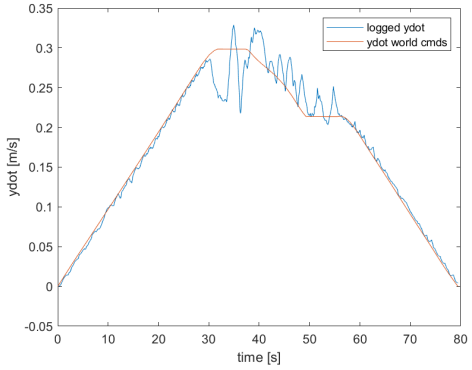
(c) Compared X



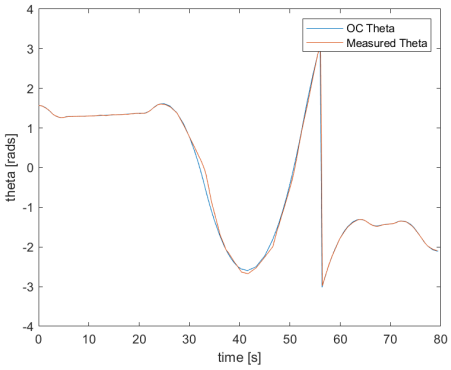
(d) Compared X Velocity



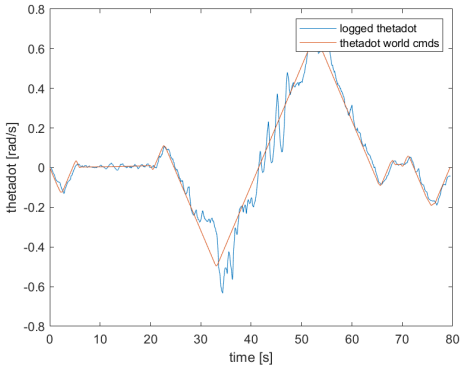
(e) Compared Y



(f) Compared Y Velocity

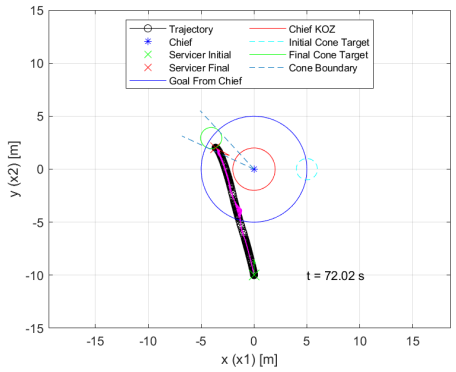


(g) Compared Theta

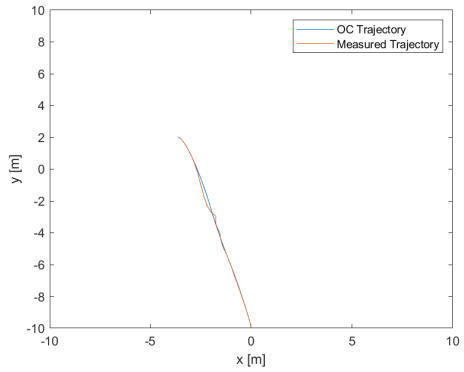


(h) Compared Theta Velocity

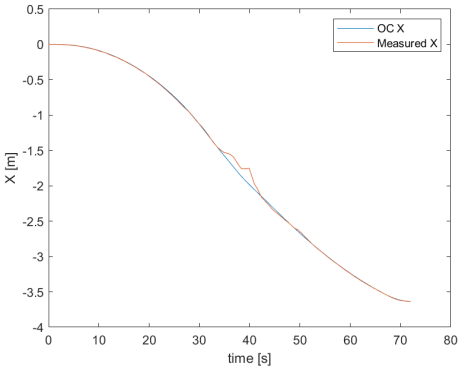
Figure 57: Test 2 MAV Lab 80% scaling, unable to perform velocities over the maximum hardware velocity.



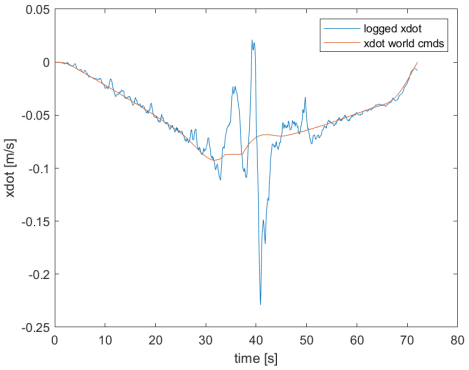
(a) Optimal Trajectory



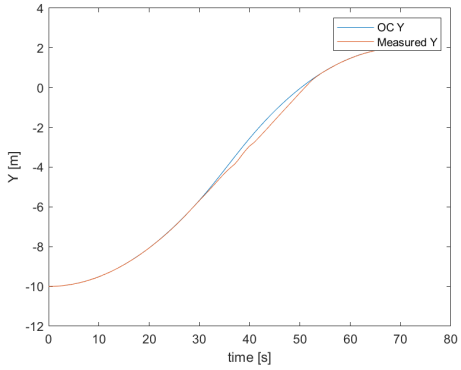
(b) Compared Trajectory



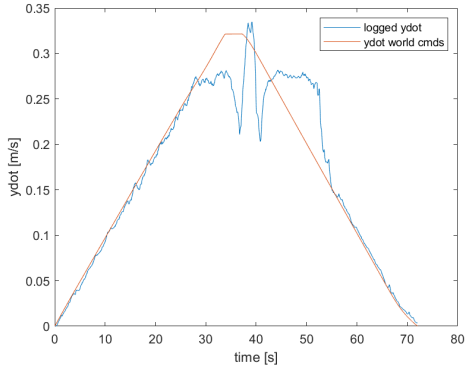
(c) Compared X



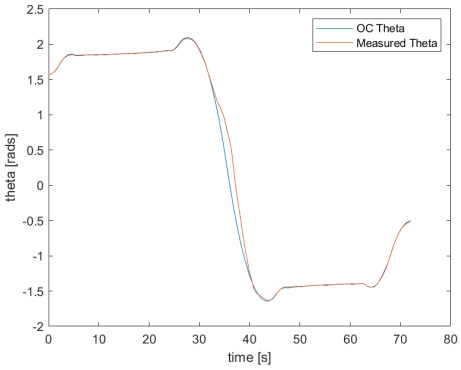
(d) Compared X Velocity



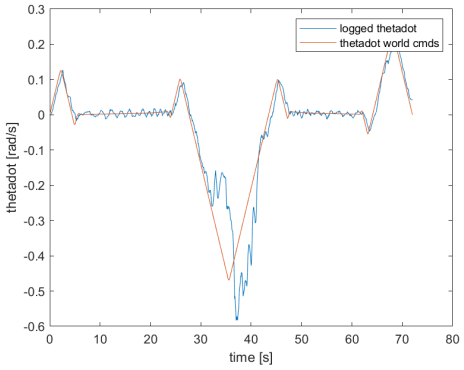
(e) Compared Y



(f) Compared Y Velocity



(g) Compared Theta



(h) Compared Theta Velocity

Figure 58: Test 3 MAV Lab 80% scaling, unable to perform velocities over the maximum hardware velocity.

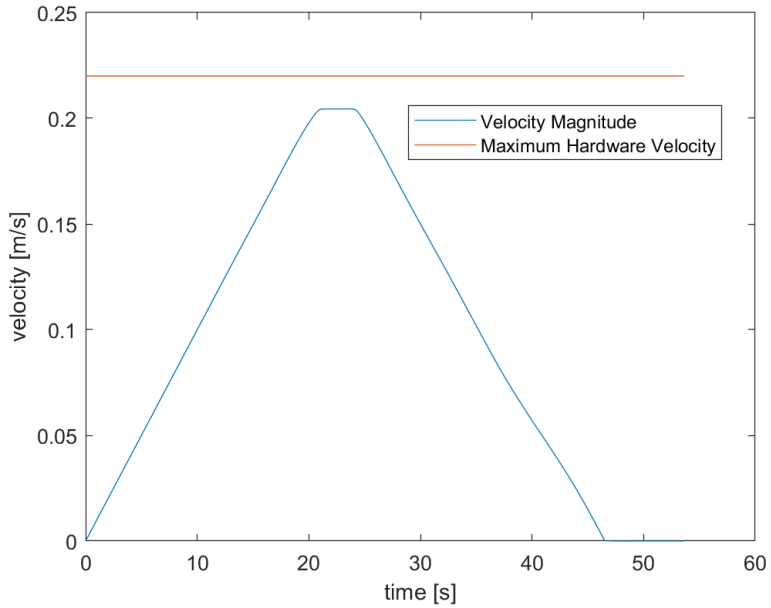


Figure 59: Test 1 under the hardware velocity limit of 0.22 m/s

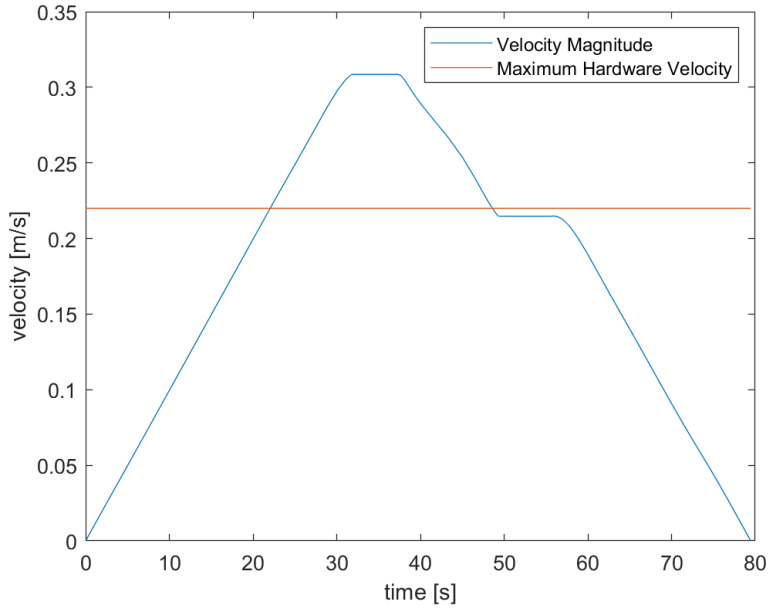


Figure 60: Test 2 without scaling requires velocities over the hardware velocity limit of 0.22 m/s. A 70% scaling reduces the peak below the limit.

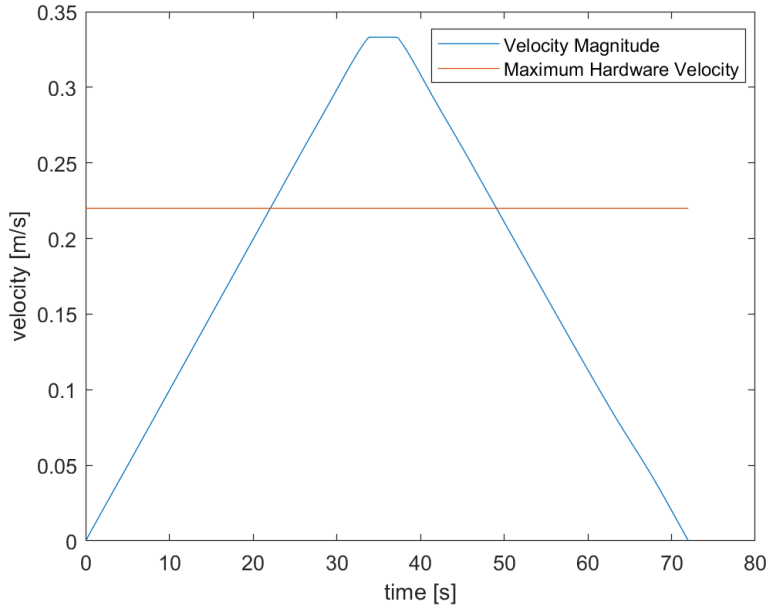
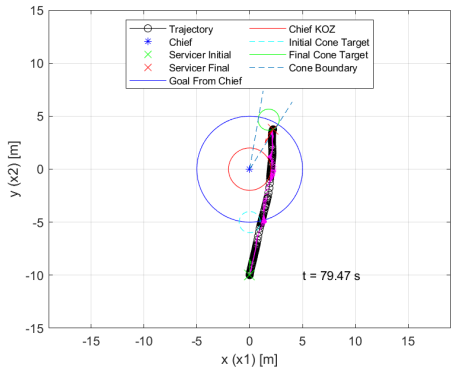
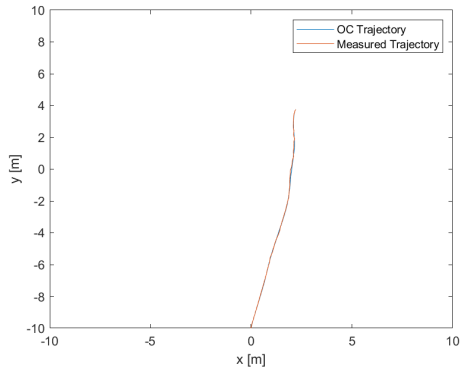


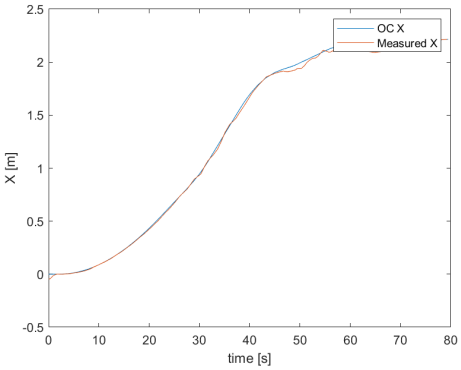
Figure 61: Test 3 without scaling requires velocities over the hardware velocity limit of 0.22 m/s. A 60% scaling reduces the peak below the limit.



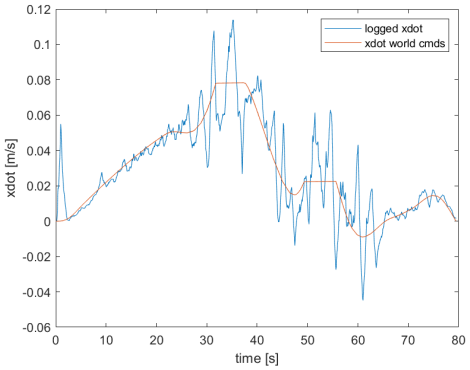
(a) Optimal Trajectory



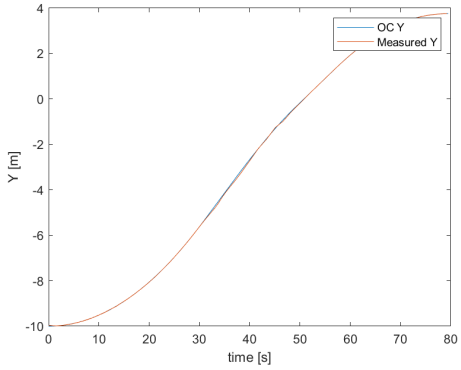
(b) Compared Trajectory



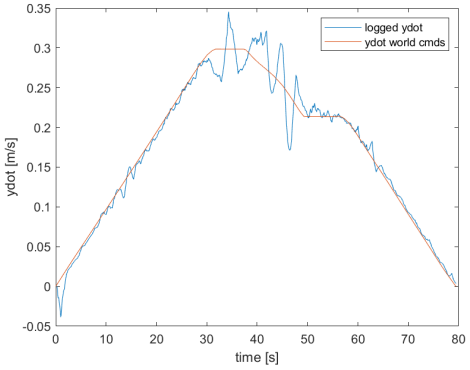
(c) Compared X



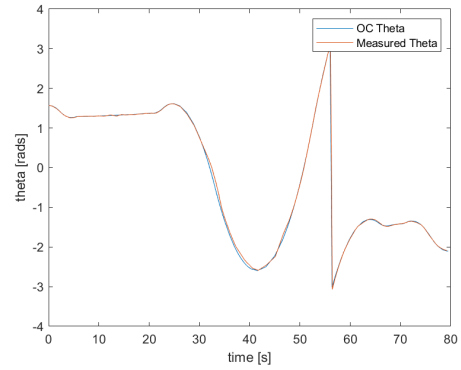
(d) Compared X Velocity



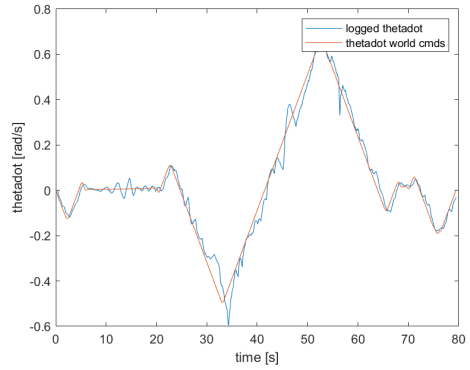
(e) Compared Y



(f) Compared Y Velocity

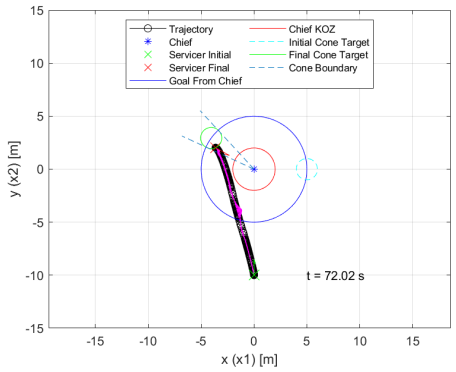


(g) Compared Theta

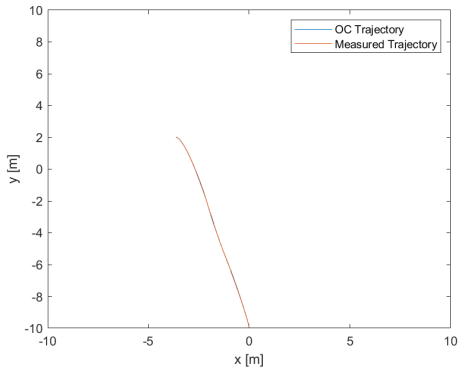


(h) Compared Theta Velocity

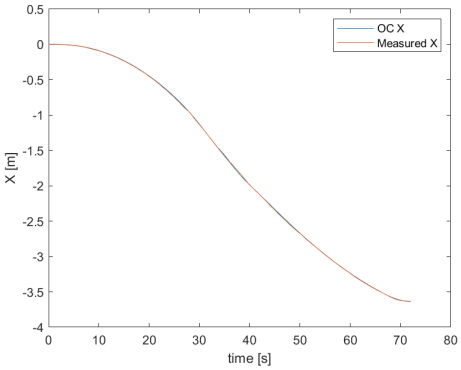
Figure 62: Test 2 MAV Lab 70% scaling, performed under maximum hardware velocity.



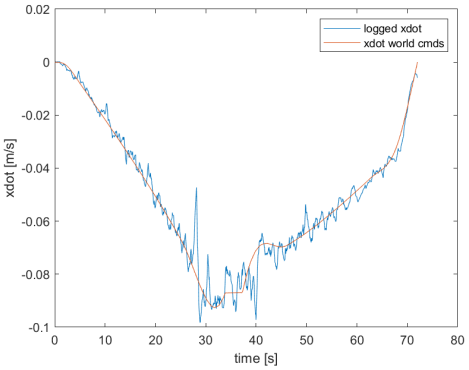
(a) Optimal Trajectory



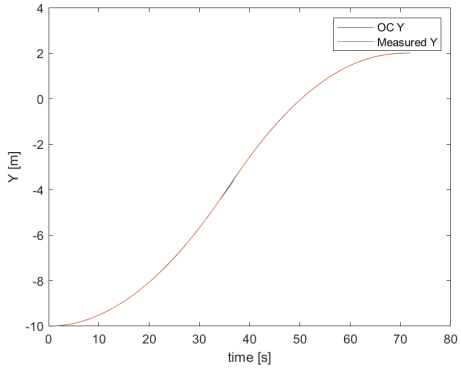
(b) Compared Trajectory



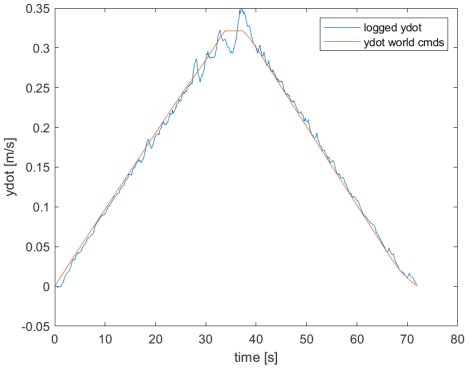
(c) Compared X



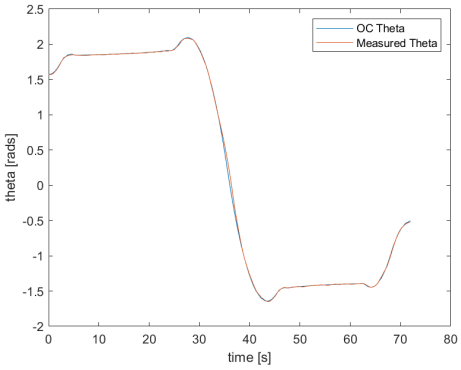
(d) Compared X Velocity



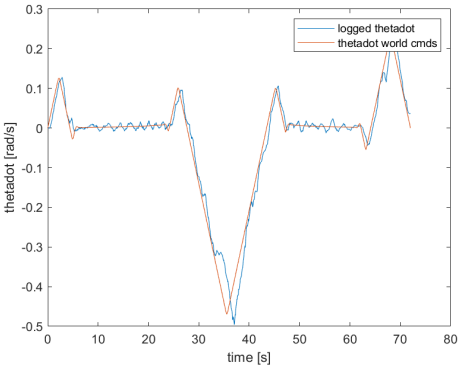
(e) Compared Y



(f) Compared Y Velocity



(g) Compared Theta



(h) Compared Theta Velocity

Figure 63: Test 3 MAV Lab 60% scaling, performed under maximum hardware velocity.

The results of this section highlight the hardware limitations of the testbed by its maximum velocity constraints. Even with ample space to perform the scenario trajectories at full scale, a degree of down scaling is still required to enable the hardware to follow the trajectory without larger errors during peak velocities. However, as shown in Figures 62 and 63, an appropriate scaling for the scenario can be implemented to take advantage of the additional space while also enabling a greater range of test scenarios.

4.4 Error Analysis

This section presents a comparison of the various test case, hardware, and controller configurations via the Root Mean Square Error (RMSE) of each. Position error (x and y) is in terms of millimeters and orientation error (θ) is in terms of radians. First, the open loop and closed loop tests on the hardware are compared to show the substantial reduction in error with feedback control. Then, the closed loop tests on the simulated and real hardware are compared to show that the hardware performed as expected from the simulation. Lastly, the error from the scaled closed loop hardware tests in the smaller VICON chamber are compared with the full scale demonstrations.

4.4.1 Open Loop vs Closed Loop

The RMSE values from each test on the omnibot hardware using the open and closed loop controllers are found in Table 7. As expected, the closed loop controller corrects for error that accumulates over the open loop demonstrations, and the RMSE values are an order of magnitude lower for all of the position errors. However, the orientation error is actually higher in a few of the closed loop cases. This is due to an unresolved issue with the feedback signal around the angle wrapping that occurs

from π to $-\pi$ in Test Cases 1, 2, and 4 as mentioned in Section 4.2.1.

Table 7: RMSE Open Loop vs Closed Loop

	Open			Closed		
RMSE	x [mm]	y [mm]	θ [rad]	x [mm]	y [mm]	θ [rad]
Test 1	83.9	71.4	0.095	5.9	8.4	0.1355
Test 2	123.6	250.8	0.0737	9.6	13.7	0.0502
Test 3	95.5	130.3	0.0526	7.2	7.5	0.0039
Test 4	168.5	128.2	0.0435	10.0	6.2	0.0813
Test 5	96.5	153.9	0.1036	5.9	8.1	0.0040
Test 6	56.7	58.0	0.0855	9.8	5.6	0.0076
Test 7	141.8	278.7	0.0877	13.9	23.4	0.0050

4.4.2 Simulation vs Hardware

The RMSE values from each test of the closed loop controller using the open_base simulation and omnibot hardware are found in Table 8. The position errors are lower in the simulation due to various errors introduced from using real hardware that are not modeled in the simulation, but the values are within the same order of magnitude. However, while the orientation errors were actually occasionally higher in the closed loop tests vs the open loop tests on the hardware, the simulated closed loop tests have higher orientation errors than the closed loop tests on the hardware in Test Cases 3, 5, 6 and 7. This is likely partially due to the simulated orientation stability issue mentioned in Section 4.2.1.

4.4.3 Full Scale vs Scaled Down

The omnibot hardware was tested in a larger VICON chamber at the MAV lab as introduced in Section 4.3 for the first three test cases. Table 9 presents the RMSE values of the closed loop controller running the hardware in the MAV lab with maximum scaling (80% for Test Case 1, 70% for Test Case 2, and 60% for Test Case 3).

Table 8: RMSE Simulation vs Hardware

	Simulation			Hardware		
RMSE	x [mm]	y [mm]	θ [rad]	x [mm]	y [mm]	θ [rad]
Test 1	1.0	2.2	0.1086	5.9	8.4	0.1355
Test 2	5.9	6.4	0.0292	9.6	13.7	0.0502
Test 3	1.5	2.0	0.0205	7.2	7.5	0.0039
Test 4	1.2	2.0	0.0283	10.0	6.2	0.0813
Test 5	1.6	2.0	0.0205	5.9	8.1	0.0040
Test 6	1.0	1.1	0.0224	9.8	5.6	0.0076
Test 7	1.8	4.2	0.0230	13.9	23.4	0.0050

Additional tests were run at 10% scaling in the MAV lab to further compare to the test environment of the 10% scaling tests in the ANT lab in Section 4.2.

The RMSE values of the tests at maximum scaling are higher than the tests at 10% scaling in Section 4.2 in the ANT lab. However, when comparing the maximum scaling tests to tests performed at 10% scaling in the same lab, they are lower. This indicates that the testbed performs the trajectories with less error while closer to the full scale dimensions, as long as the required velocities are below the hardware limits. If the velocities are not scaled down below the hardware limits, performance is degraded as shown in Figures 57 and 58. The source of the additional error in the 10% scaling tests in the MAV lab compared to the same tests performed in the ANT lab was not thoroughly investigated. However, it is likely that the error could be due to an outdated motion capture facility calibration as the larger facility was not recalibrated prior to these tests.

Table 9: RMSE Full Scale vs Scaled Down

	Max Scaling (MAV)			10% Scaling (MAV)			10% Scaling (ANT)		
RMSE	x [mm]	y [mm]	θ [rad]	x [mm]	y [mm]	θ [rad]	x [mm]	y [mm]	θ [rad]
Test 1	9.2	3.2	0.2201	36.1	52.9	0.2904	5.9	8.4	0.1355
Test 2	19.0	28.8	0.0839	23.5	34.7	0.0716	9.6	13.7	0.0502
Test 3	7.2	12.9	0.0347	28.8	25.7	0.0081	7.2	7.5	0.0039

4.5 Chapter Summary

This chapter analyzes the performance of the omnibot testbed through simulation and hardware across seven test cases. An overview of the test scenario and a graphical representation of the trajectory constraints and terminal conditions is shown in Section 4.1. Highlights of the results are discussed in Section 4.2, however all of the test results for the four runs of each test case can be found in the appendices. While the tests in Section 4.2 are performed at 10% scaling due to lab space constraints, tests performed in a larger lab space are presented in Section 4.3 to compare results from tests at closer to full scale. Section 4.4 then presents the root-mean-square error (RMSE) of each test case to compare open loop to closed loop performance, simulation to hardware performance, and full scale to 10% scaling performance.

V. Conclusions

This thesis presents the background, design, implementation, and results of the omnibot mobile robot testbed for autonomous satellite servicing research. The omnibot offers a three degrees of freedom (3-DOF) hardware platform built from a commercially available robotics kit and does not require dedicated lab space. Demonstrations can be performed within existing motion capture facilities commonly being used for indoor unmanned aerial vehicle (UAV) or mobile robotics research. The omnibot platform serves as a lower-fidelity hardware testbed that helps fill the gap between simulation and limited higher-fidelity hardware demonstration.

The background on the kinematics and dynamics that reconcile the holonomic motion of the mobile robot to the behavior of a deputy satellite on orbit is found in Sections 2.1 and 2.2. Section 2.3 introduces optimal control methods and a commercial solver used to numerically solve for reference trajectories for the hardware testbed to demonstrate. Section 2.4 gives an overview of the Robot Operating System (ROS), which is used for the testbed network framework.

Section 3.1 details the various components of the omnibot hardware, as well as the motion capture system that measures the performance of the testbed. Section 3.2 outlines the ROS network setup, including the computing environment that provides the supporting functions to run and record the demonstrations. Section 3.3 provides an example satellite servicing scenario and the optimal control formulation used to compute the guidance trajectories to complete an inspection of a rotating chief satellite in minimum time. Section 3.3 also provides an example feedback control algorithm used to follow the guidance trajectories.

The results of demonstrating the example inspection scenario with the hardware testbed are presented and analyzed in Sections 4.1 and 4.2. The orbital servicing demonstration scenarios take place over a 10 m^2 area, but due to space constraints in

the primary lab used in this research the demonstrations in Section 4.2 are scaled down to a 1 m^2 area (10% scaling). Results from demonstrations over an 8 m^2 area (80% scaling) in a larger motion capture facility are presented in Section 4.3. Section 4.4 then analyzes the root-mean-square error (RMSE) of the omnibot pose throughout each test case to compare the performance of the testbed across the various test conditions.

5.1 Thesis Contributions

- Built a low-cost 3-DOF hardware testbed for autonomous satellite servicing research based on a variant of the commercially available TurtleBot mobile robot. The testbed provides a lower-fidelity alternative to limited state-of-the-art facilities for rapid control algorithm prototyping.

- Set up a ROS network that enables communication between the mobile robot, the satellite control simulation that outputs the guidance trajectories and control commands, and the motion capture system that relays navigation measurements for feedback control and testbed performance analysis.

- Provided an example satellite servicing scenario involving the inspection of a rotating chief satellite that demonstrates the capabilities of the testbed. The example scenario and example feedback controller are designed to be modified or replaced to aid the development of future controls research.

- Performed simulation and hardware testing to validate the performance of the testbed across seven test cases based on the example scenario. Testing is performed in two different motion capture facilities to highlight the flexibility of the testbed in different environments and to show the effect of distance scaling in smaller laboratories.

5.2 Future Work

- Build additional copies of the omnibot mobile robot to enable demonstration of multi-agent behavior. An additional omnibot could be introduced to model the chief satellite instead of only using the origin of the Local Vertical, Local Horizontal (LVLH) reference frame.

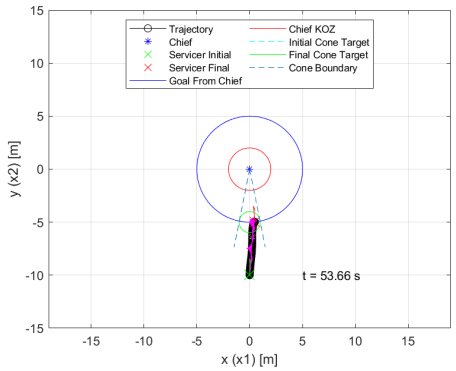
- Implement the on-board laser distance sensor to enable a state estimation component of the research with the testbed. Higher fidelity odometry could be implemented to improve on-board navigation and reduce the reliance on the motion capture system.

- Consider adding more phases to the satellite servicing scenario. Run tests cases with different initial conditions, deputy satellite configurations, chief satellite rotation rates, terminal conditions, optimal control objective functions, and time frames.

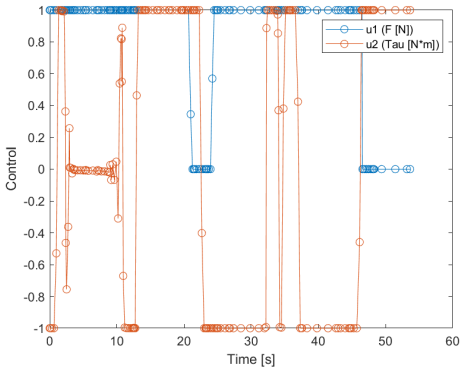
- Analyze the performance of more robust control algorithms or state-of-the-art techniques such as Artificial Potential Functions (APF) or Model Predictive Control (MPC) [17].

Appendix A. Test Case 1

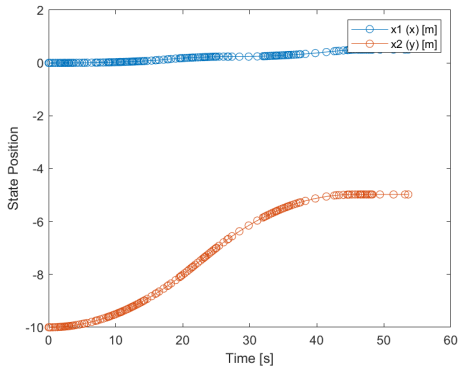
Appendix A includes all of the output figures for Test Case 1. Analysis of these results is found in Section 4.2.1. Test Case 1 uses Clohessy-Wiltshire dynamics and a stationary chief with no rotation rate and an initial orientation of $-\pi/2$ radians.



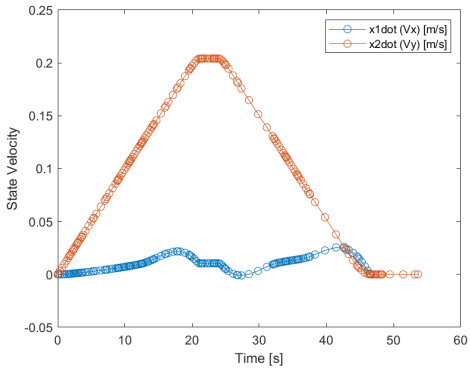
(a) Trajectory



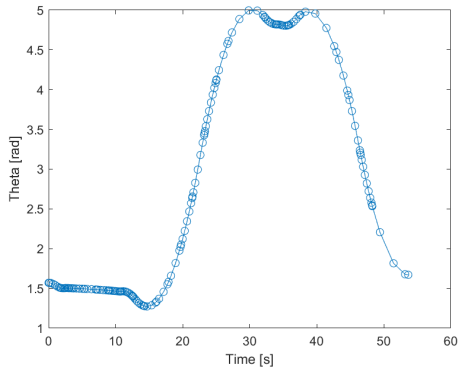
(b) Control History



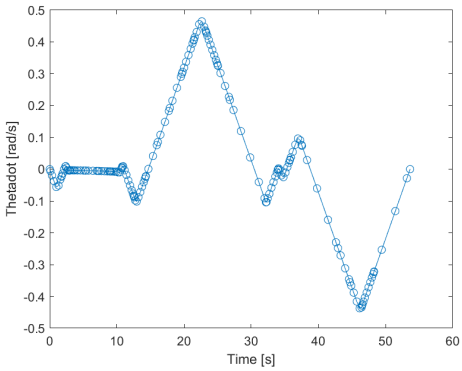
(c) Position



(d) Planar Velocities

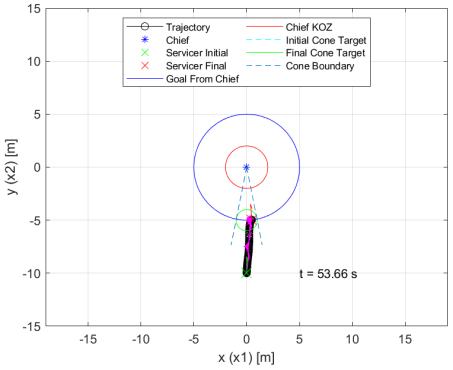


(e) Orientation

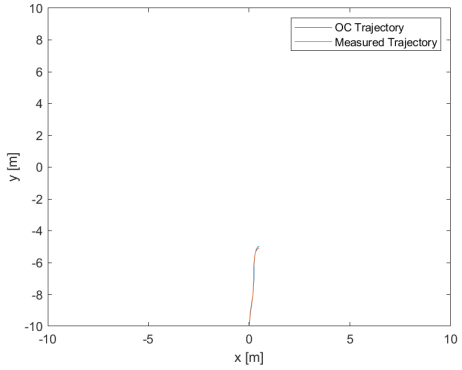


(f) Angular Velocity

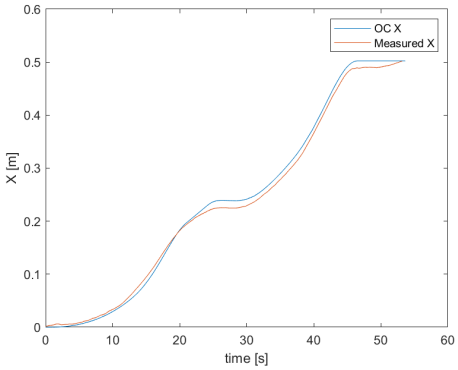
Figure A1: Test 1 Time Optimal Solution



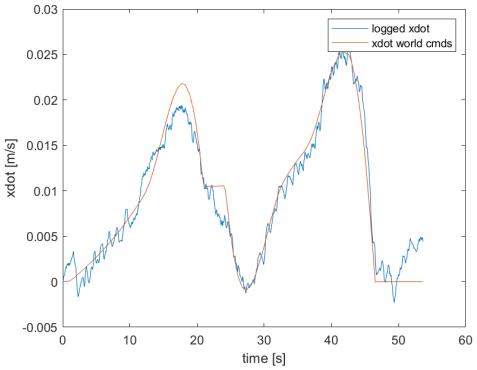
(a) Optimal Trajectory



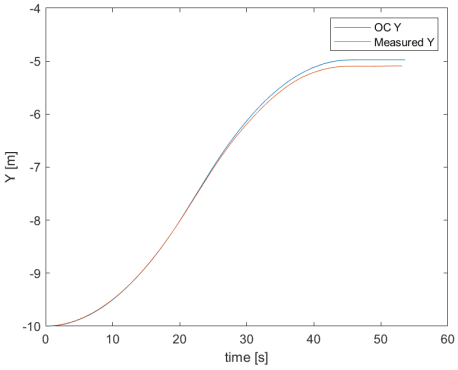
(b) Compared Trajectory



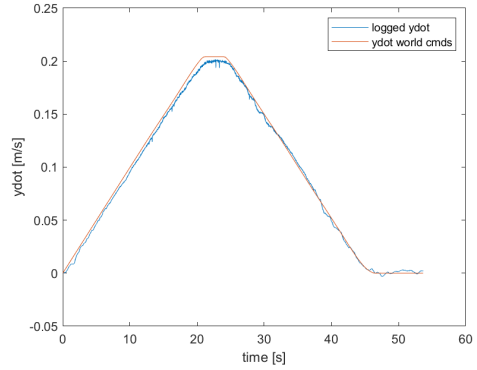
(c) Compared X



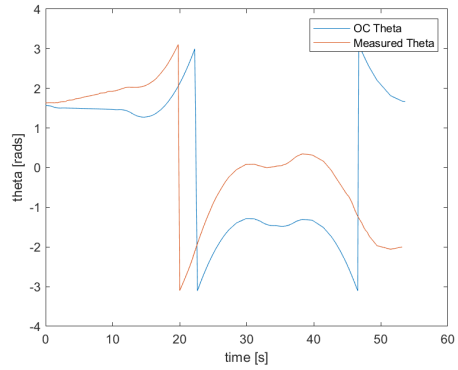
(d) Compared X Velocity



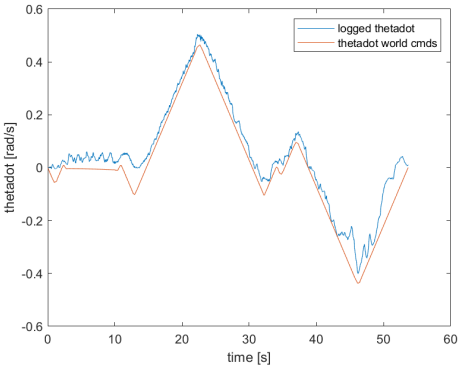
(e) Compared Y



(f) Compared Y Velocity

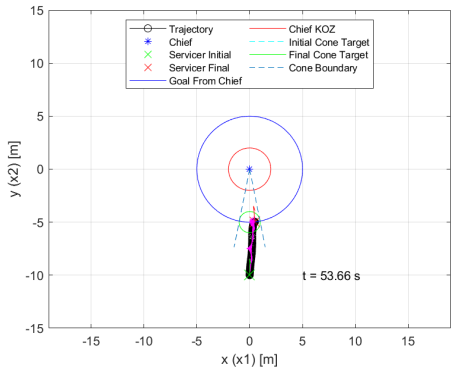


(g) Compared Theta

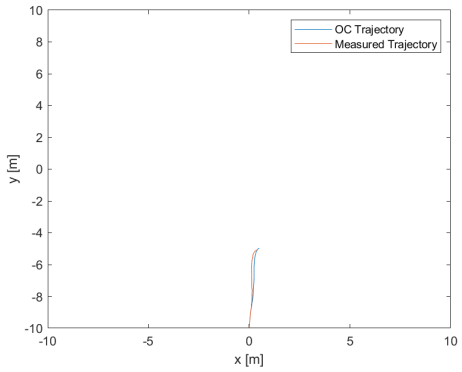


(h) Compared Theta Velocity

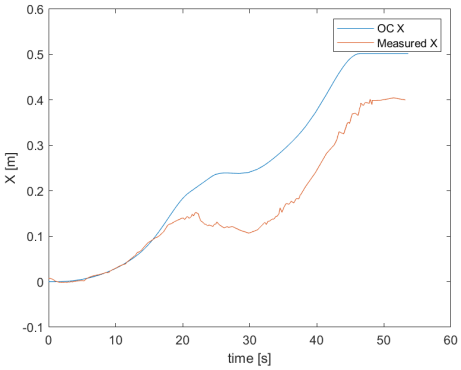
Figure A2: Test 1 Simulated Open Loop Control



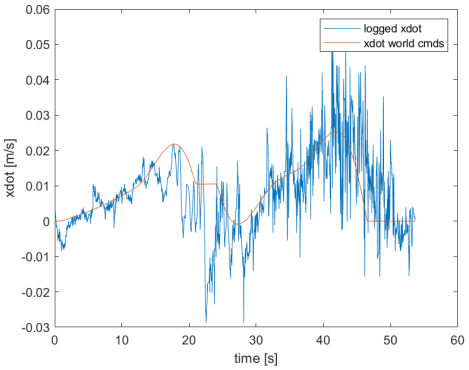
(a) Optimal Trajectory



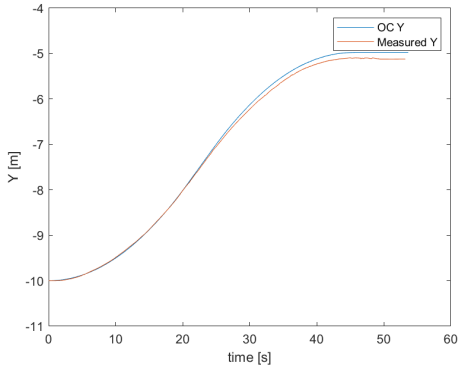
(b) Compared Trajectory



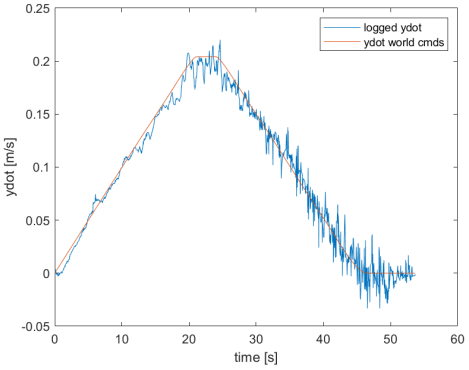
(c) Compared X



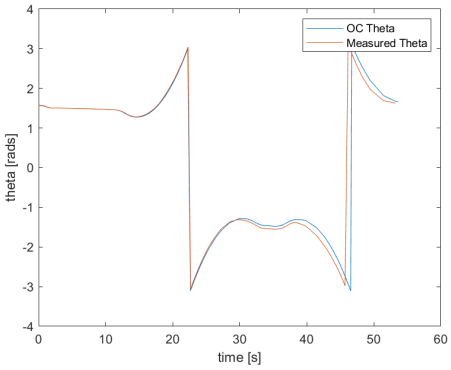
(d) Compared X Velocity



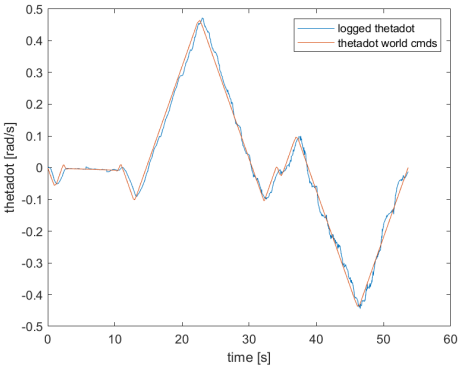
(e) Compared Y



(f) Compared Y Velocity

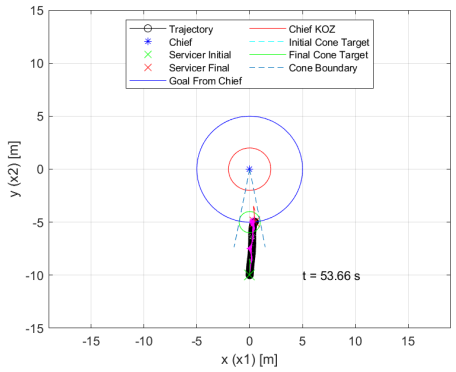


(g) Compared Theta

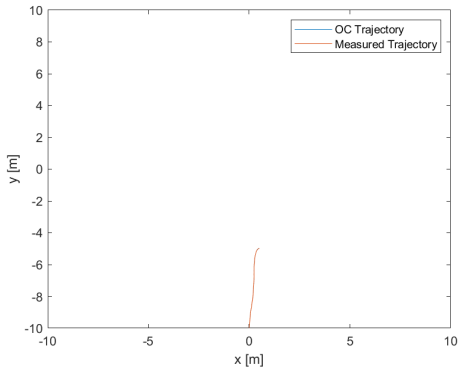


(h) Compared Theta Velocity

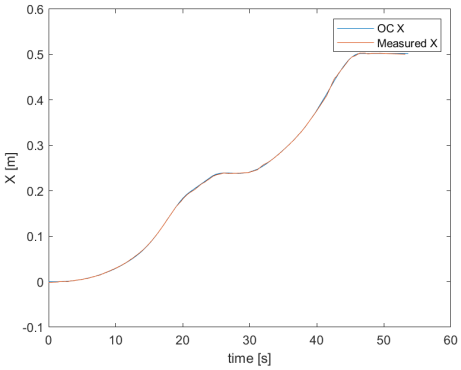
Figure A3: Test 1 Hardware Open Loop Control



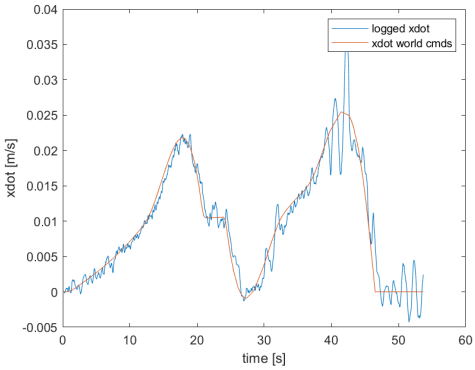
(a) Optimal Trajectory



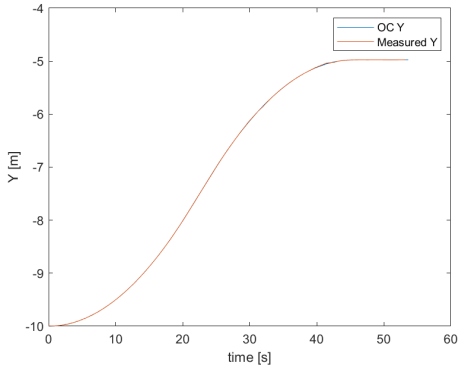
(b) Compared Trajectory



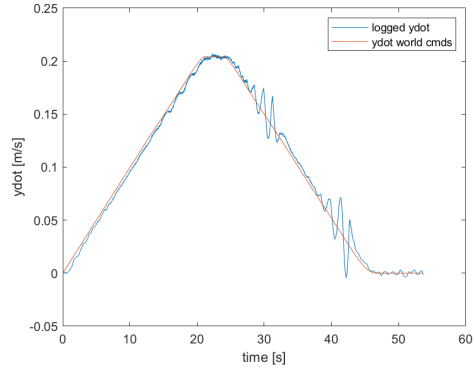
(c) Compared X



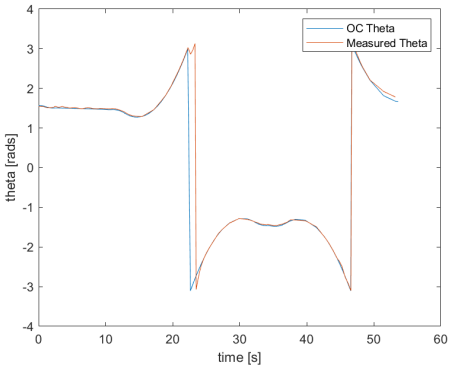
(d) Compared X Velocity



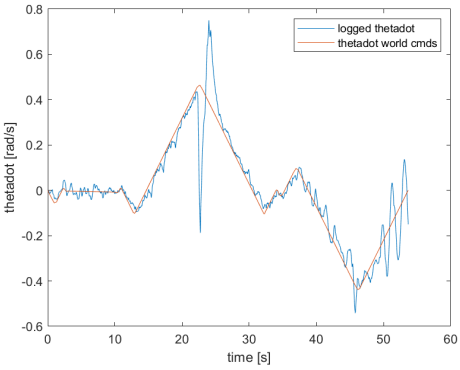
(e) Compared Y



(f) Compared Y Velocity

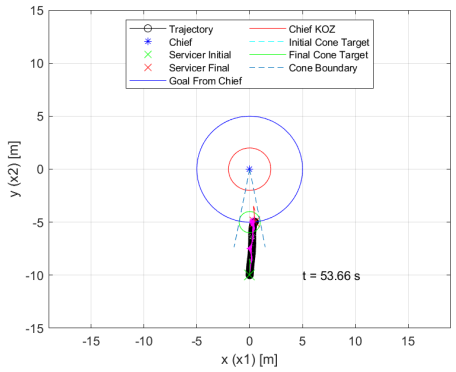


(g) Compared Theta

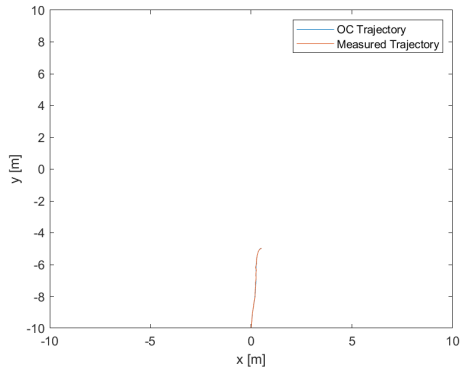


(h) Compared Theta Velocity

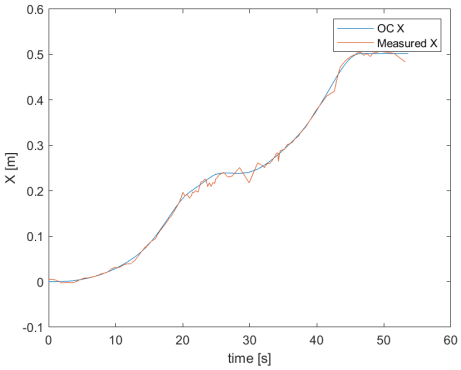
Figure A4: Test 1 Simulated Closed Loop Control



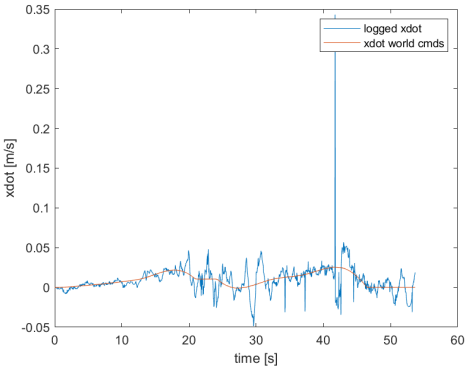
(a) Optimal Trajectory



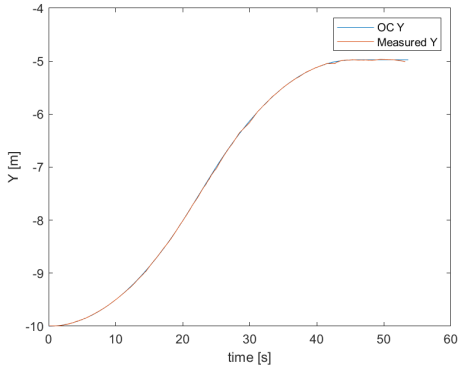
(b) Compared Trajectory



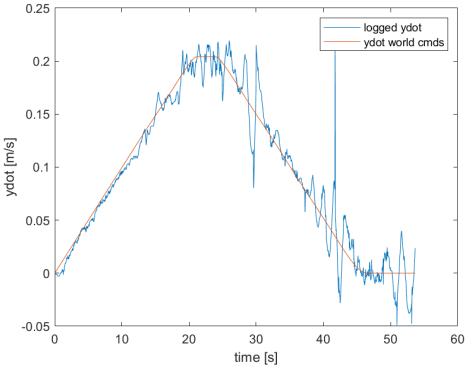
(c) Compared X



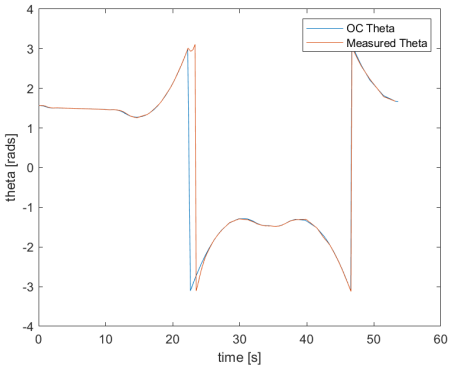
(d) Compared X Velocity



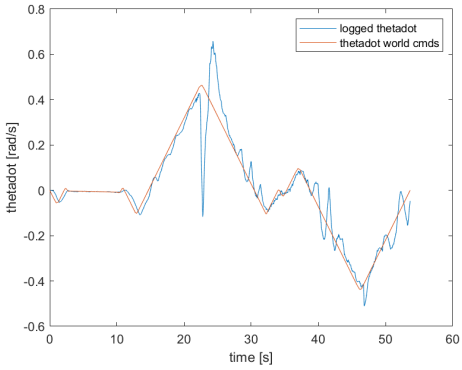
(e) Compared Y



(f) Compared Y Velocity



(g) Compared Theta

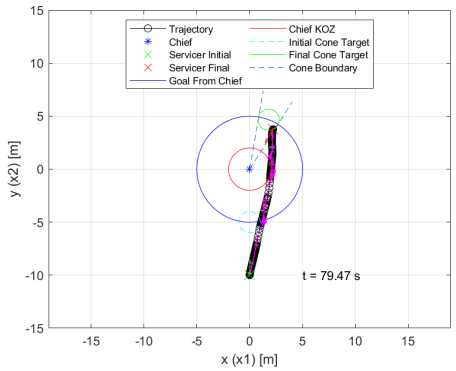


(h) Compared Theta Velocity

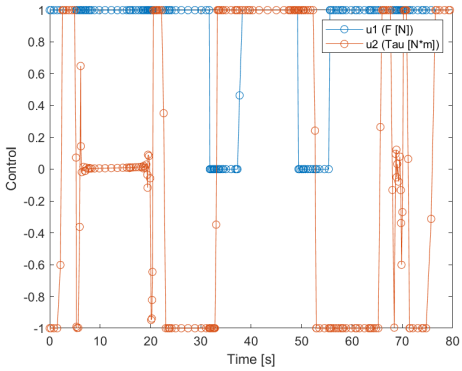
Figure A5: Test 1 Hardware Closed Loop Control

Appendix B. Test Case 2

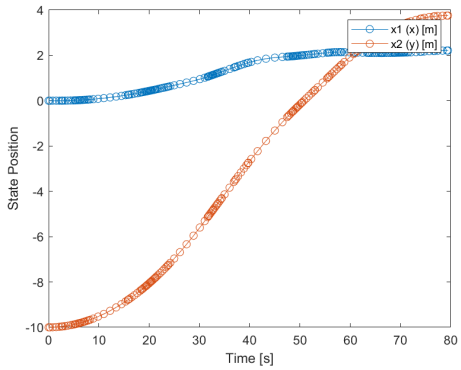
Appendix B includes all of the output figures for Test Case 2. Analysis of these results is found in Section 4.2.2. Test Case 2 uses Clohessy-Wiltshire dynamics and a rotating chief with a 2 deg/s rotation rate and an initial orientation of $-\pi/2$ radians.



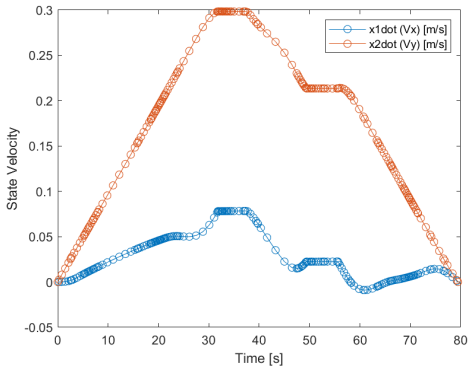
(a) Trajectory



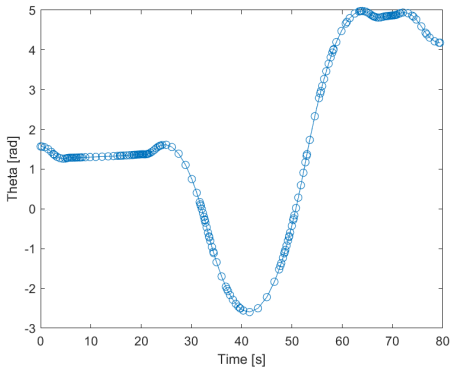
(b) Control History



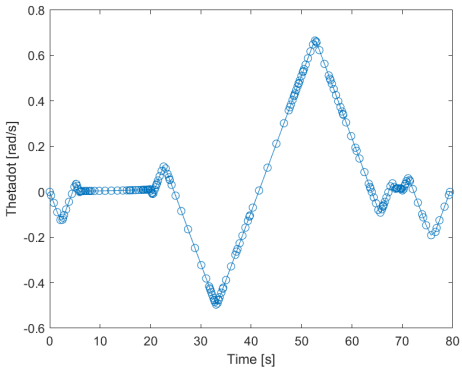
(c) Position



(d) Planar Velocities

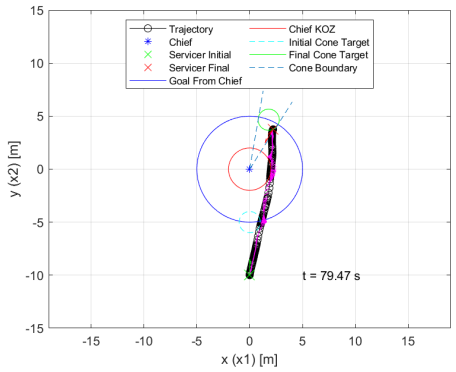


(e) Orientation

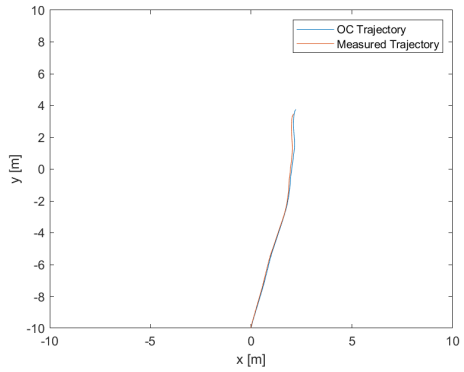


(f) Angular Velocity

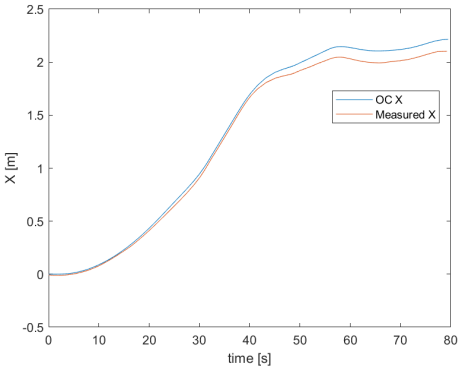
Figure B1: Test 2 Time Optimal Solution



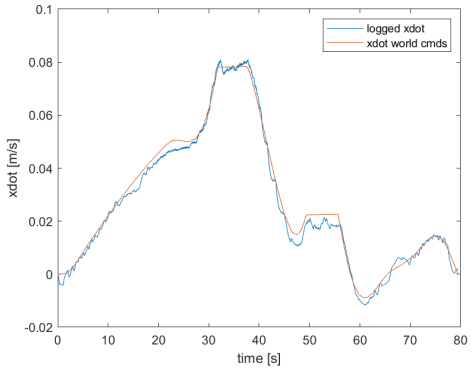
(a) Optimal Trajectory



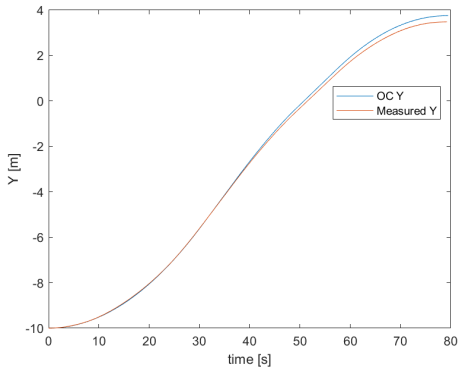
(b) Compared Trajectory



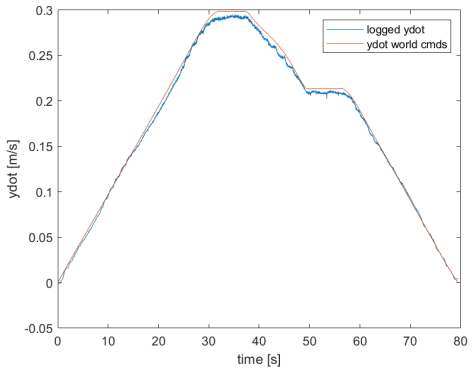
(c) Compared X



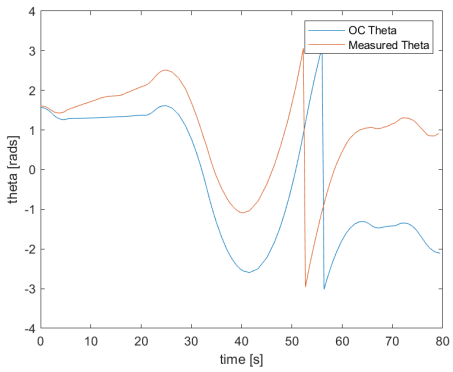
(d) Compared X Velocity



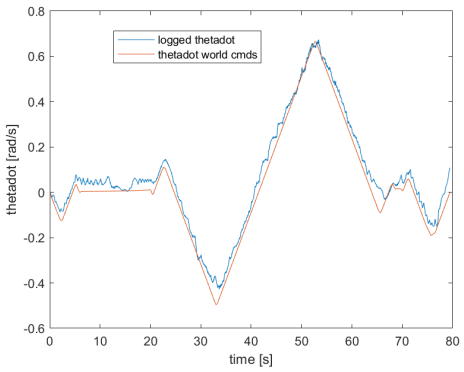
(e) Compared Y



(f) Compared Y Velocity

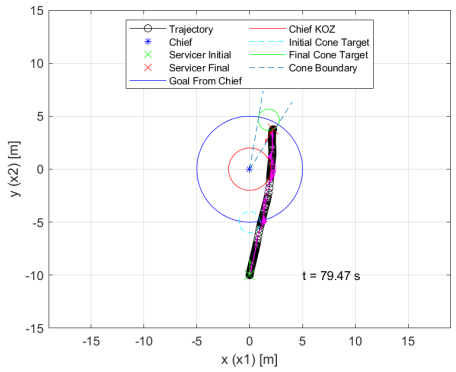


(g) Compared Theta

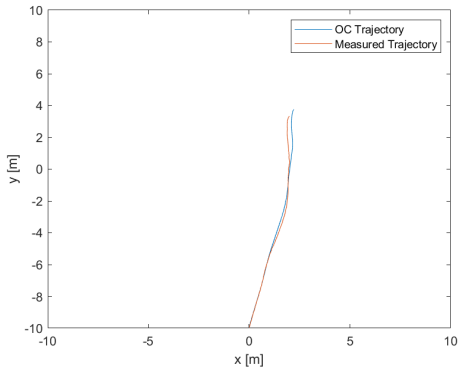


(h) Compared Theta Velocity

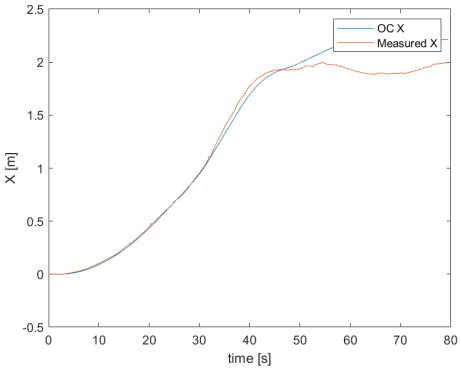
Figure B2: Test 2 Simulated Open Loop Control



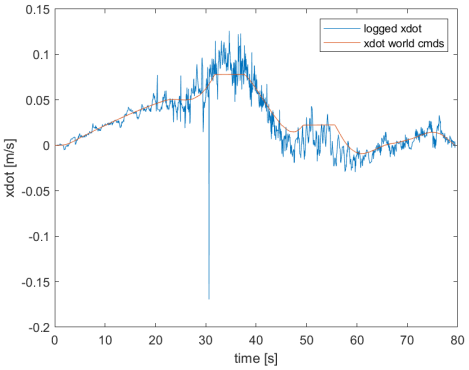
(a) Optimal Trajectory



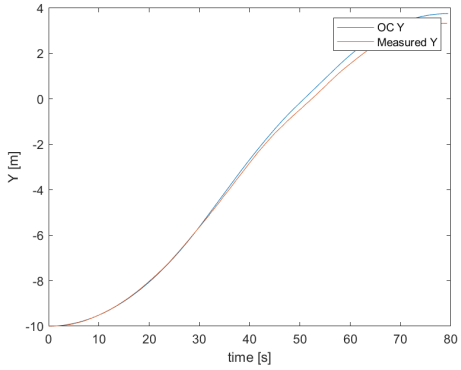
(b) Compared Trajectory



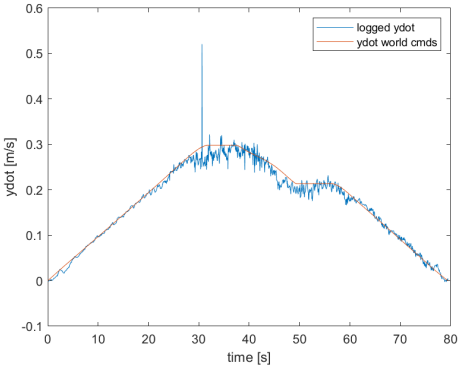
(c) Compared X



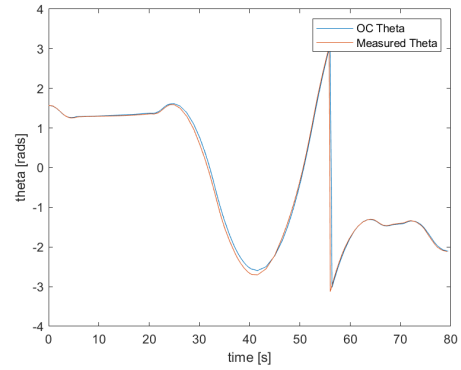
(d) Compared X Velocity



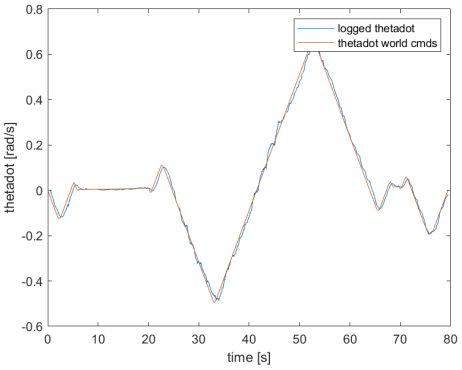
(e) Compared Y



(f) Compared Y Velocity

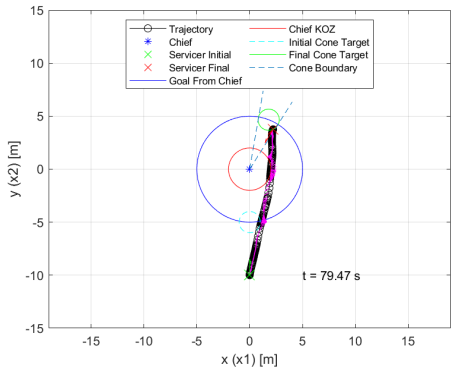


(g) Compared Theta

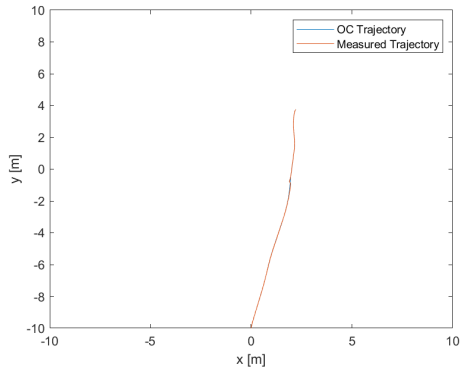


(h) Compared Theta Velocity

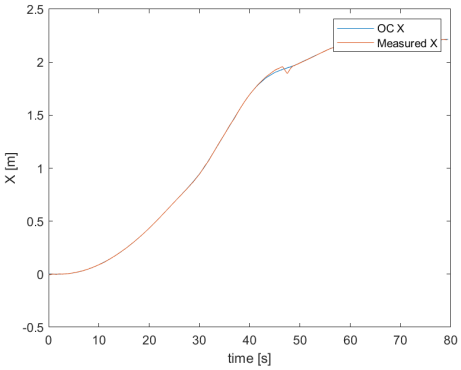
Figure B3: Test 2 Hardware Open Loop Control



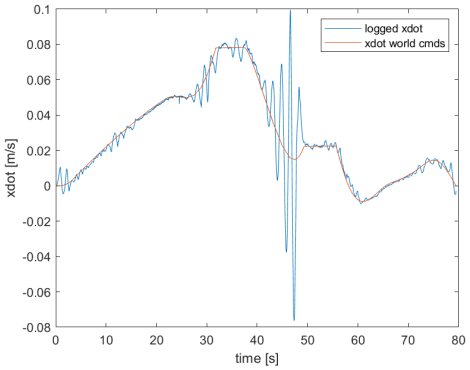
(a) Optimal Trajectory



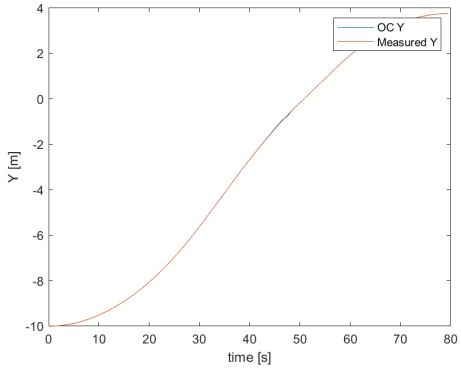
(b) Compared Trajectory



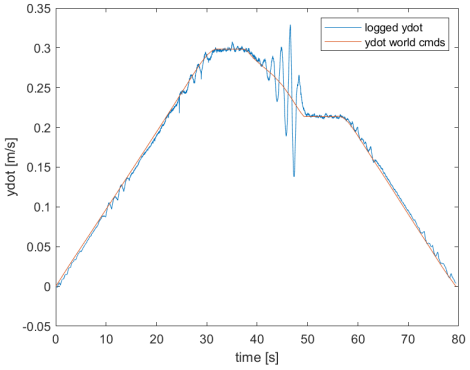
(c) Compared X



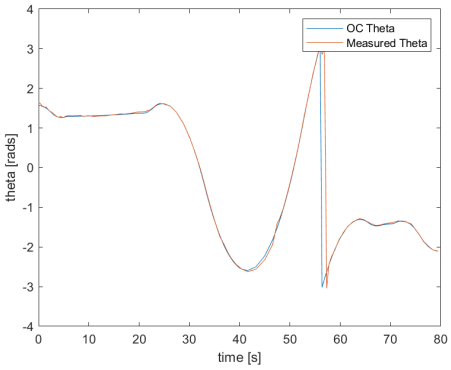
(d) Compared X Velocity



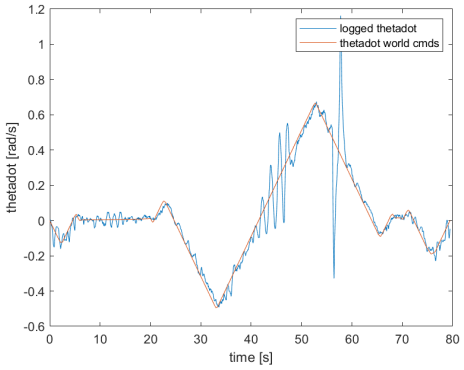
(e) Compared Y



(f) Compared Y Velocity

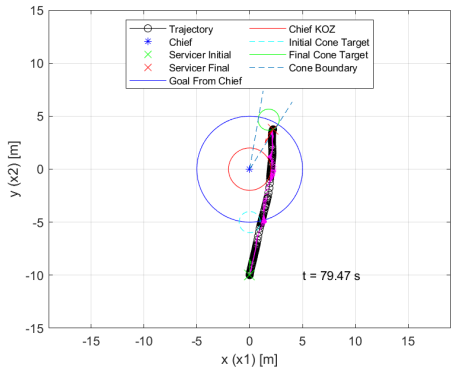


(g) Compared Theta

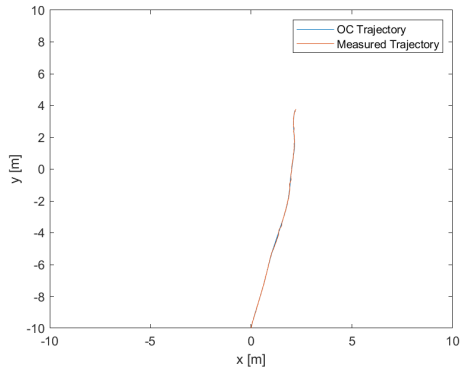


(h) Compared Theta Velocity

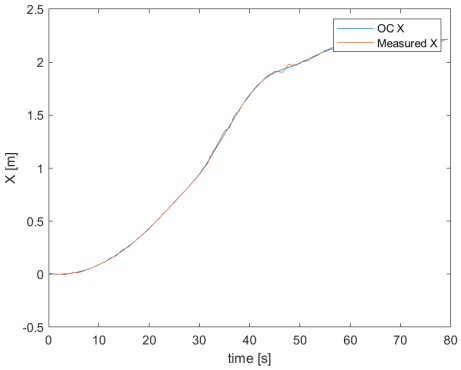
Figure B4: Test 2 Simulated Closed Loop Control



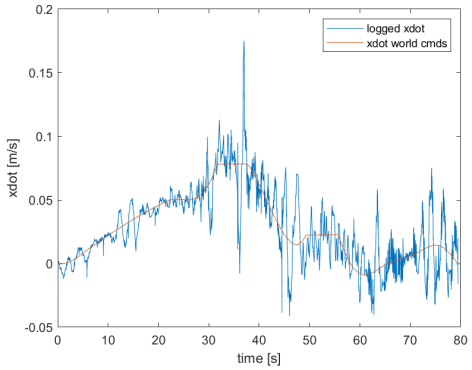
(a) Optimal Trajectory



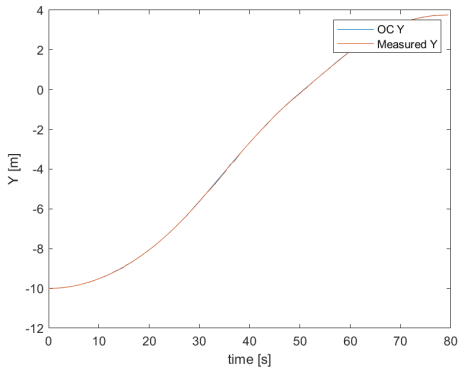
(b) Compared Trajectory



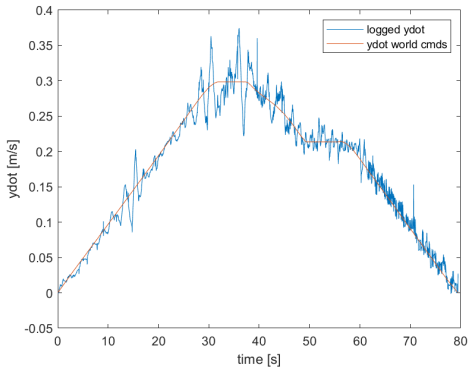
(c) Compared X



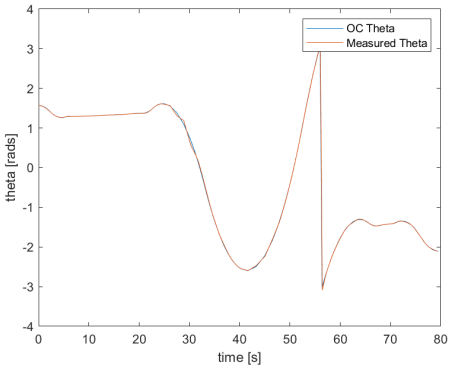
(d) Compared X Velocity



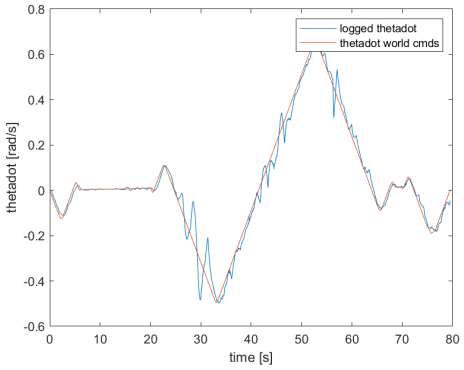
(e) Compared Y



(f) Compared Y Velocity



(g) Compared Theta

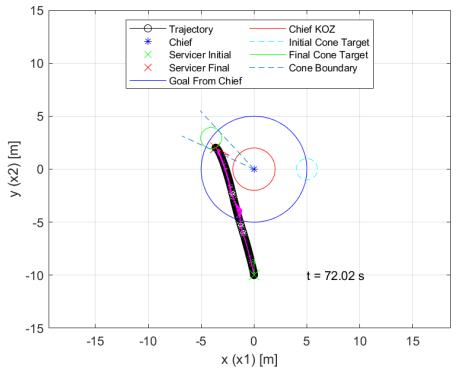


(h) Compared Theta Velocity

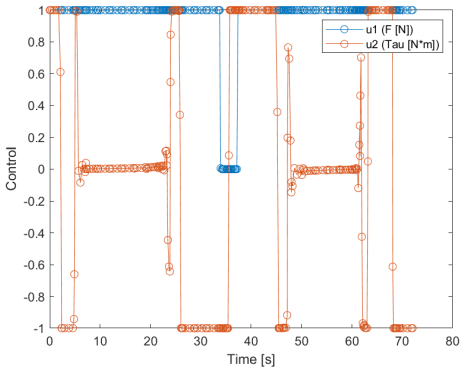
Figure B5: Test 2 Hardware Closed Loop Control

Appendix C. Test Case 3

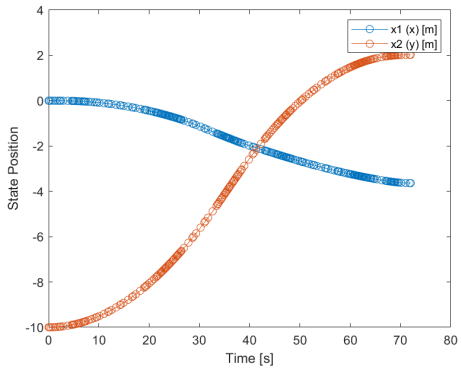
Appendix C includes all of the output figures for Test Case 3. Analysis of these results is found in Section 4.2.3. Test Case 3 uses Clohessy-Wiltshire dynamics and a rotating chief with a 2 deg/s rotation rate and an initial orientation of 0 radians.



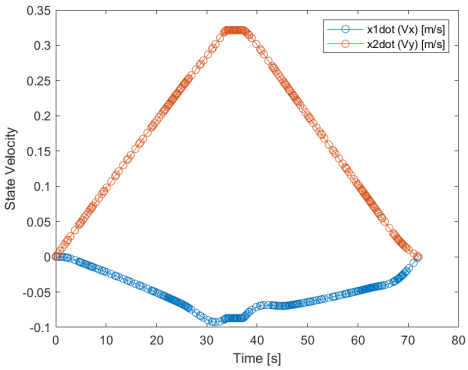
(a) Trajectory



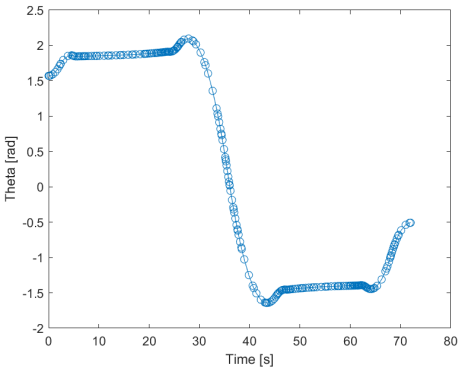
(b) Control History



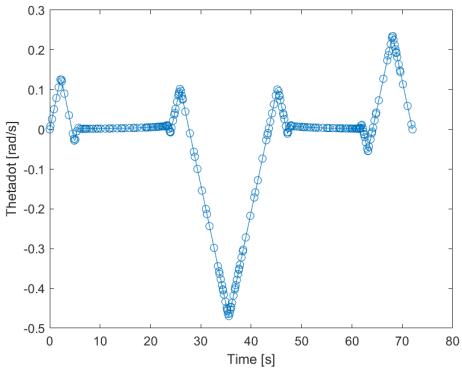
(c) Position



(d) Planar Velocities

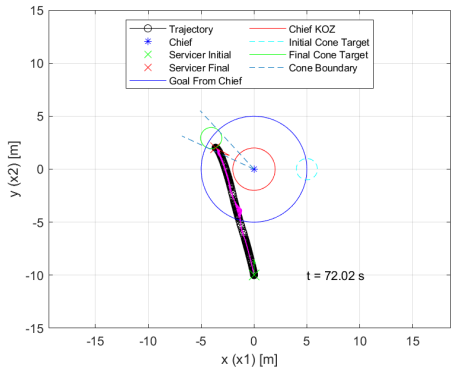


(e) Orientation

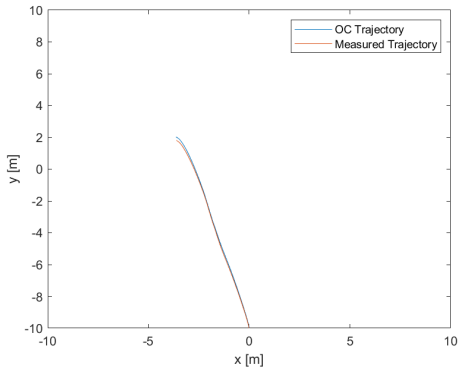


(f) Angular Velocity

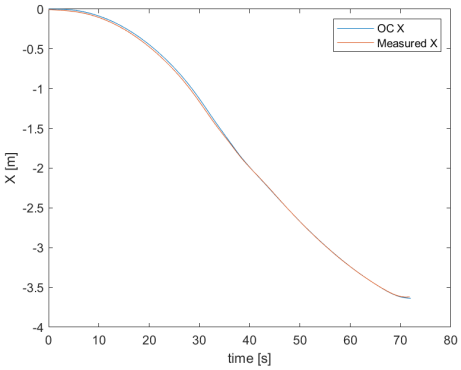
Figure C1: Test 3 Time Optimal Solution



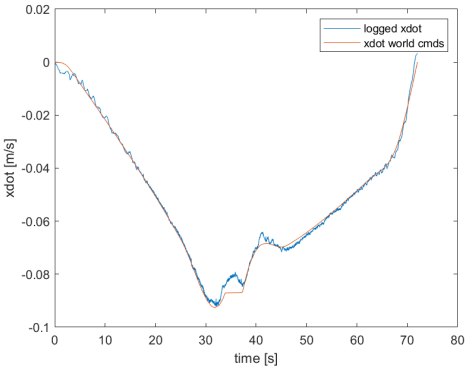
(a) Optimal Trajectory



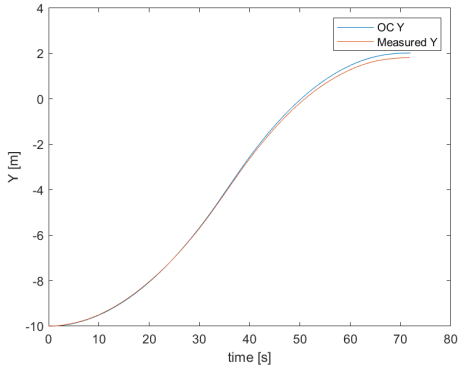
(b) Compared Trajectory



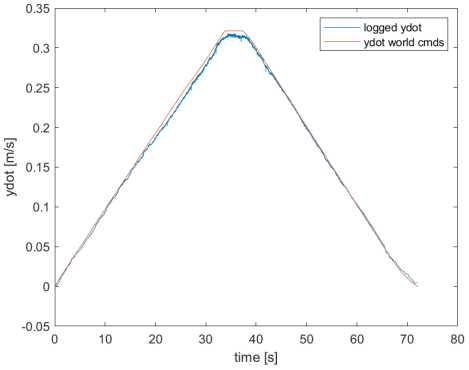
(c) Compared X



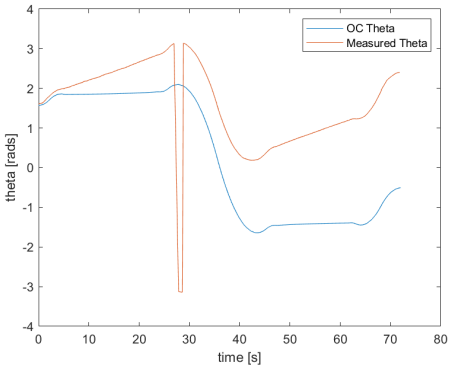
(d) Compared X Velocity



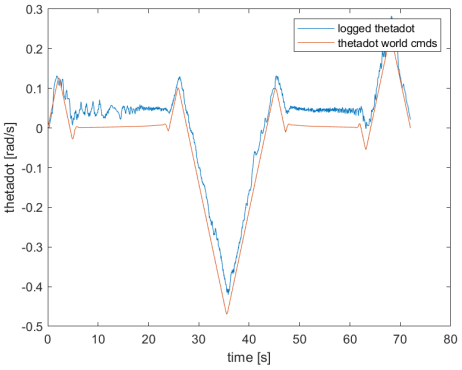
(e) Compared Y



(f) Compared Y Velocity

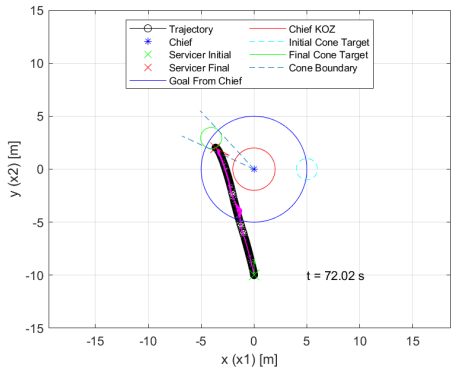


(g) Compared Theta

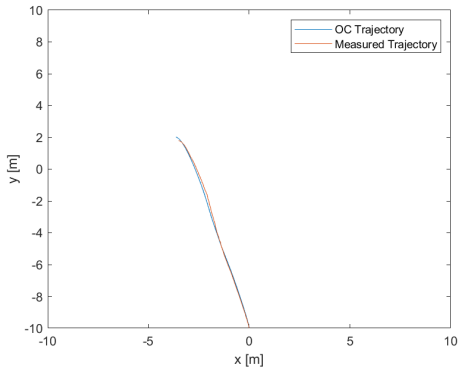


(h) Compared Theta Velocity

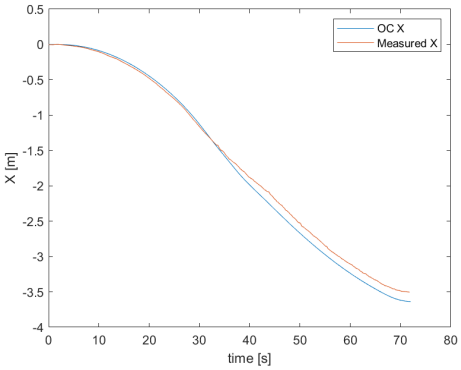
Figure C2: Test 3 Simulated Open Loop Control



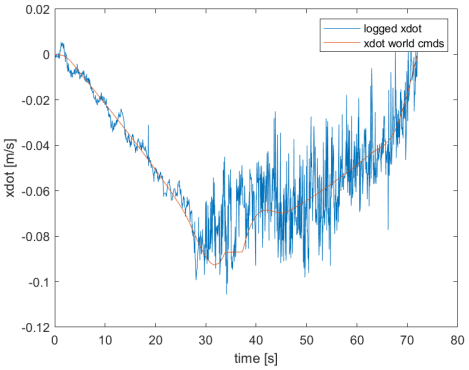
(a) Optimal Trajectory



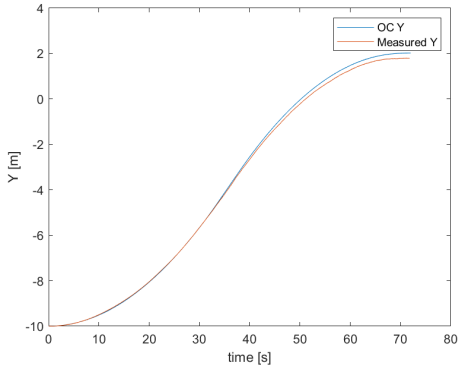
(b) Compared Trajectory



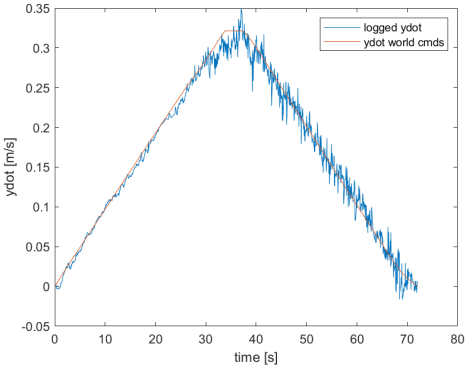
(c) Compared X



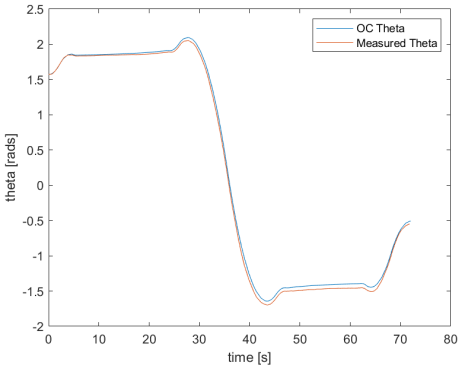
(d) Compared X Velocity



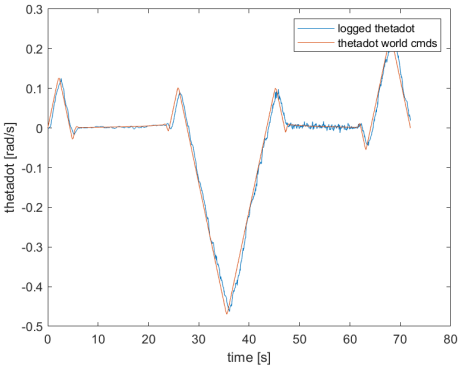
(e) Compared Y



(f) Compared Y Velocity

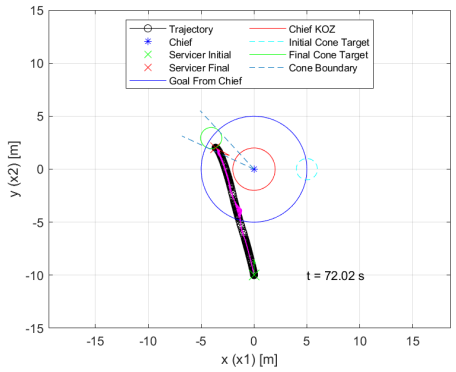


(g) Compared Theta

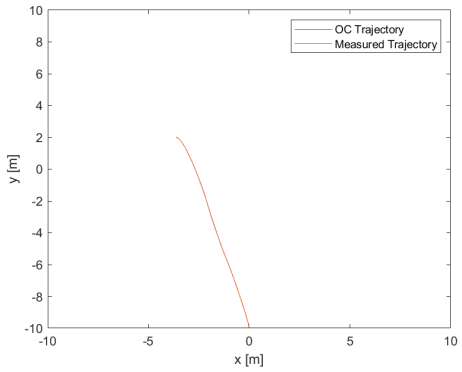


(h) Compared Theta Velocity

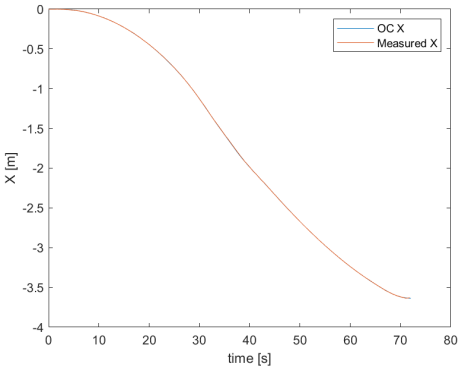
Figure C3: Test 3 Hardware Open Loop Control



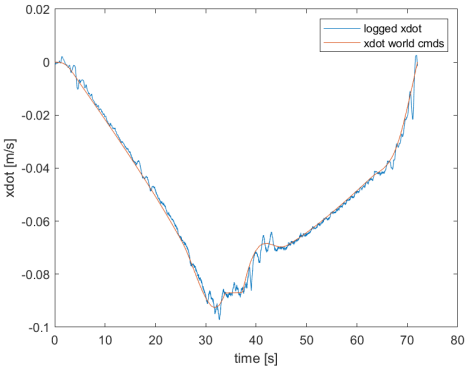
(a) Optimal Trajectory



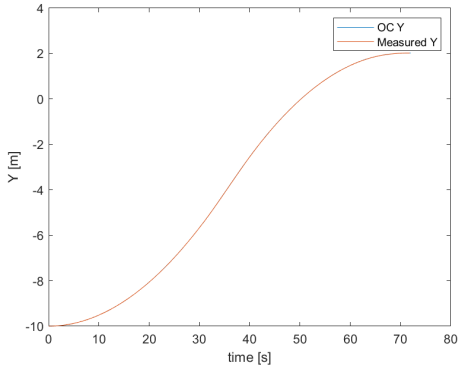
(b) Compared Trajectory



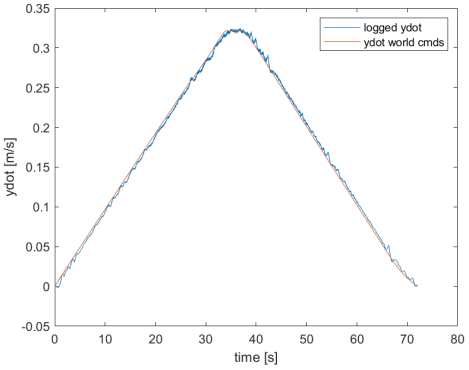
(c) Compared X



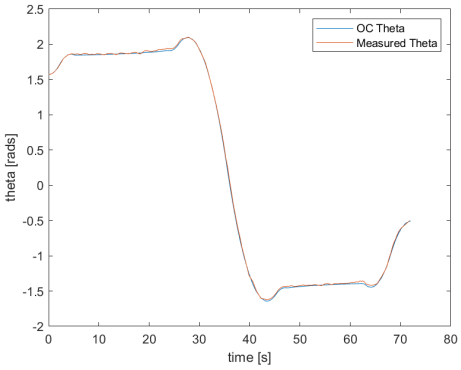
(d) Compared X Velocity



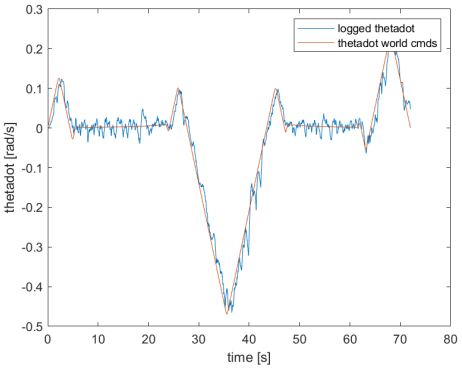
(e) Compared Y



(f) Compared Y Velocity

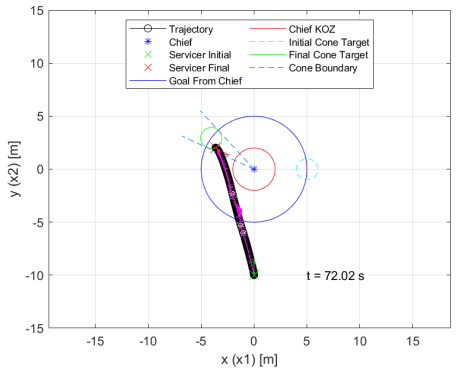


(g) Compared Theta

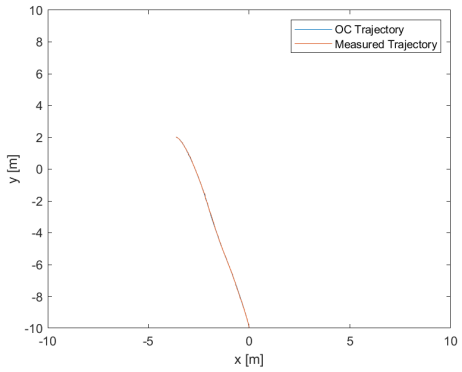


(h) Compared Theta Velocity

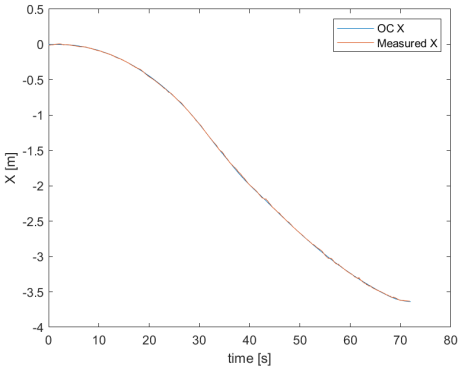
Figure C4: Test 3 Simulated Closed Loop Control



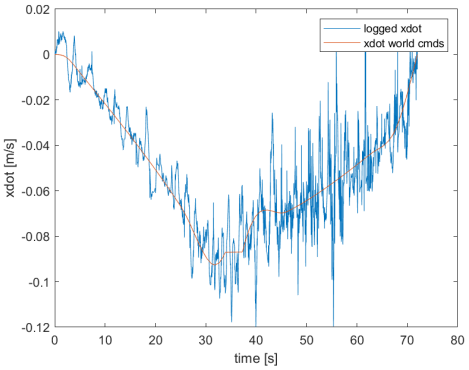
(a) Optimal Trajectory



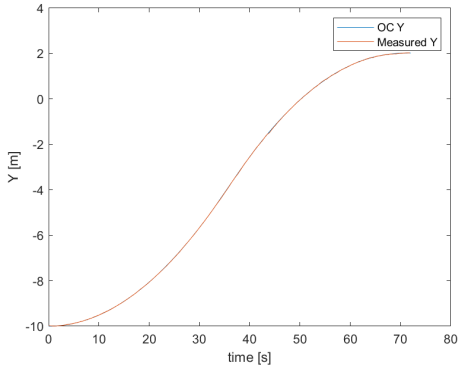
(b) Compared Trajectory



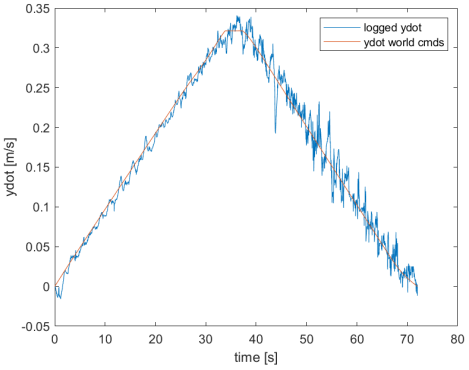
(c) Compared X



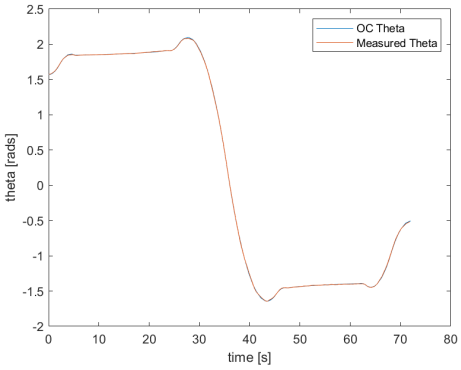
(d) Compared X Velocity



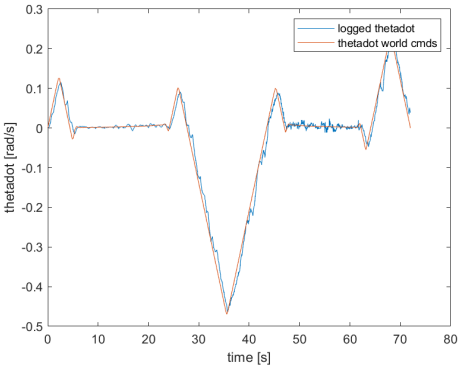
(e) Compared Y



(f) Compared Y Velocity



(g) Compared Theta

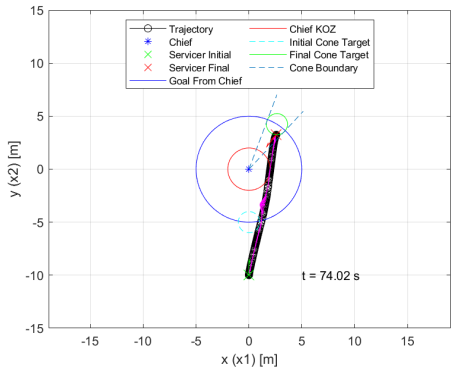


(h) Compared Theta Velocity

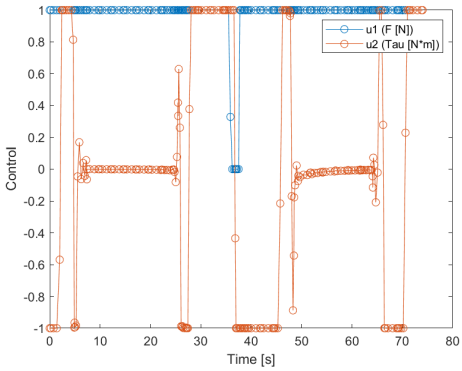
Figure C5: Test 3 Hardware Closed Loop Control

Appendix D. Test Case 4

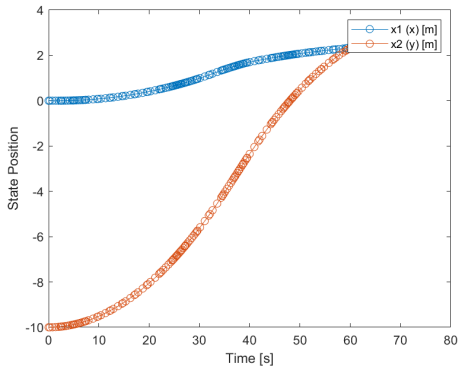
Appendix D includes all of the output figures for Test Case 4. Analysis of these results is found in Section 4.2.4. Test Case 4 uses Double Integrator dynamics and a rotating chief with a 2 deg/s rotation rate and an initial orientation of $-\pi/2$ radians.



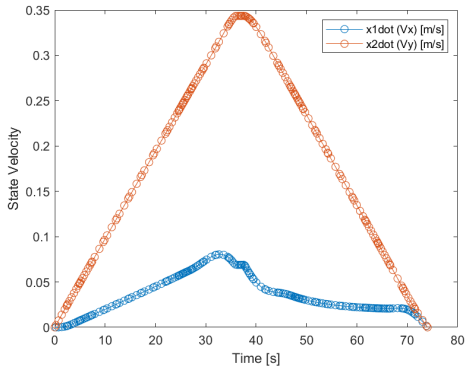
(a) Trajectory



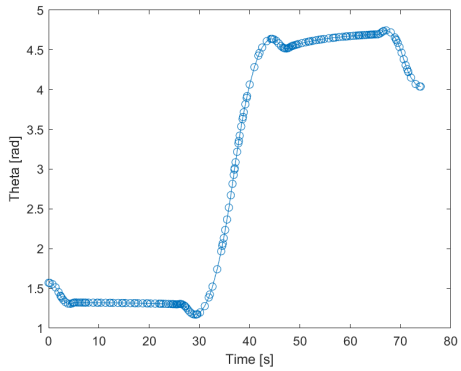
(b) Control History



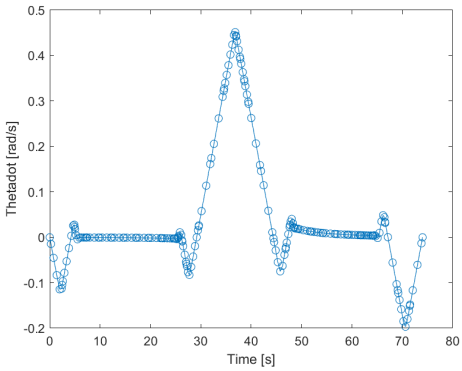
(c) Position



(d) Planar Velocities

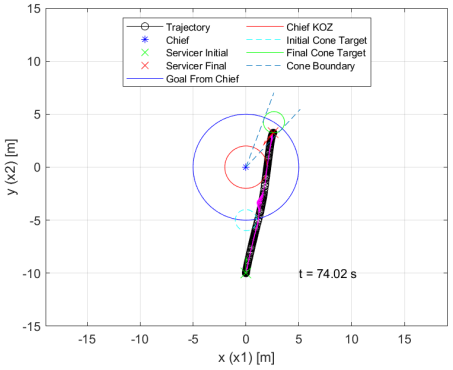


(e) Orientation

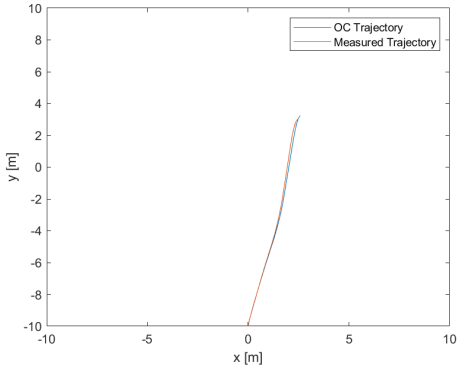


(f) Angular Velocity

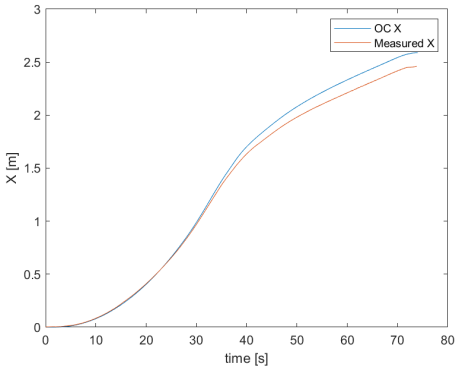
Figure D1: Test 4 Time Optimal Solution



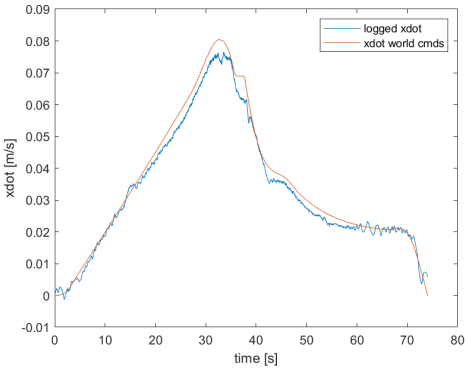
(a) Optimal Trajectory



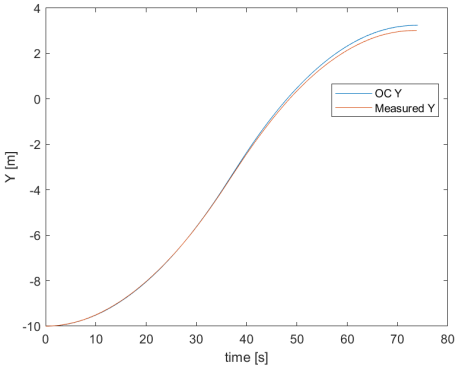
(b) Compared Trajectory



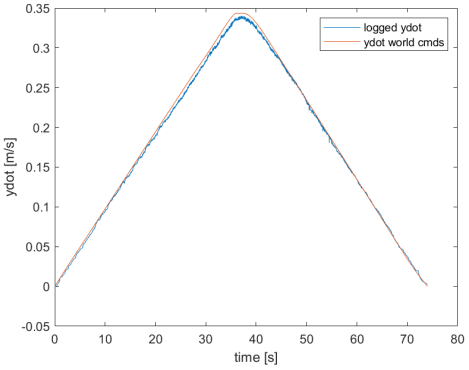
(c) Compared X



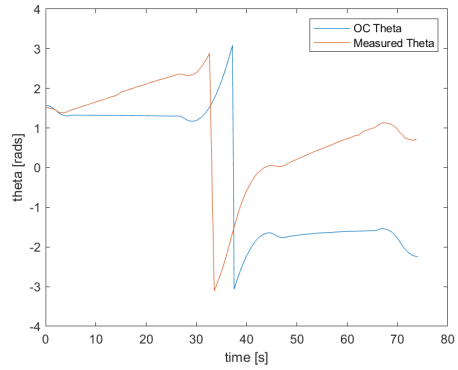
(d) Compared X Velocity



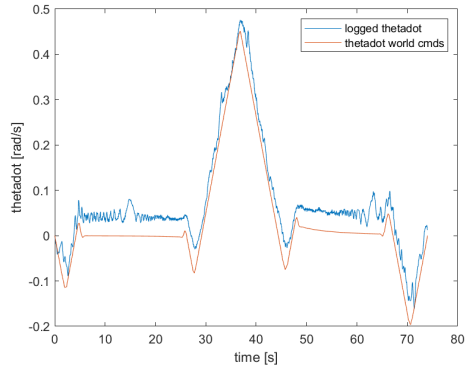
(e) Compared Y



(f) Compared Y Velocity

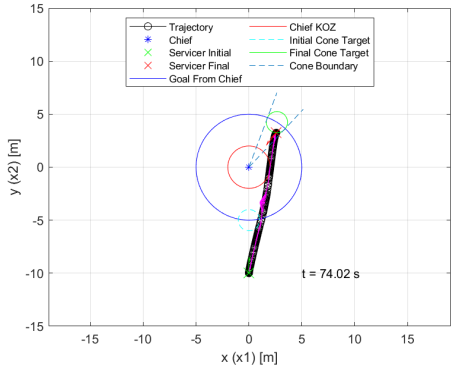


(g) Compared Theta

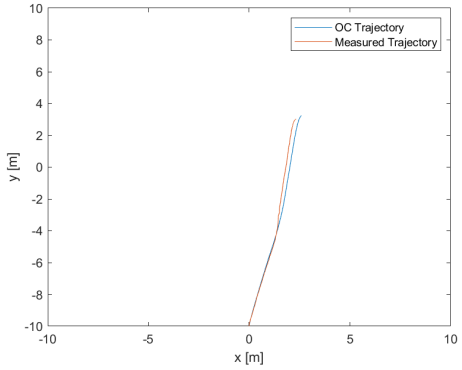


(h) Compared Theta Velocity

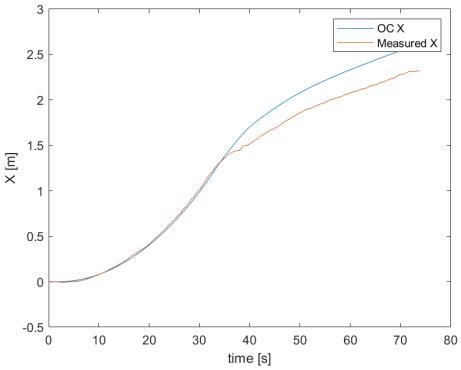
Figure D2: Test 4 Simulated Open Loop Control



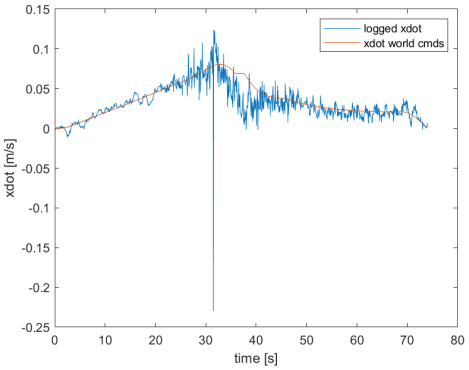
(a) Optimal Trajectory



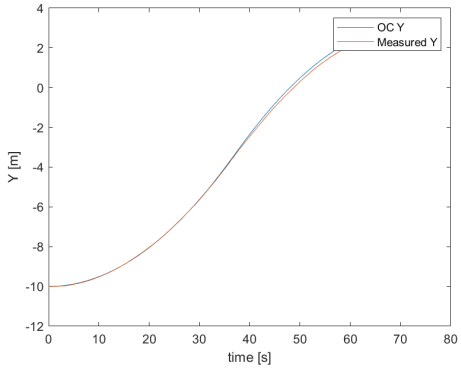
(b) Compared Trajectory



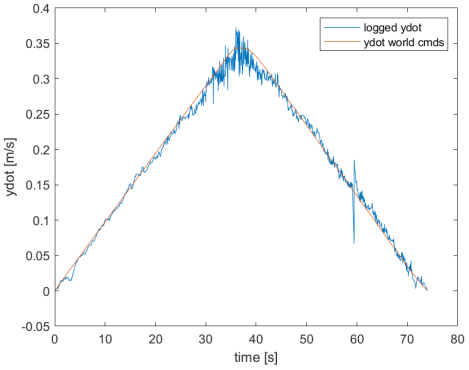
(c) Compared X



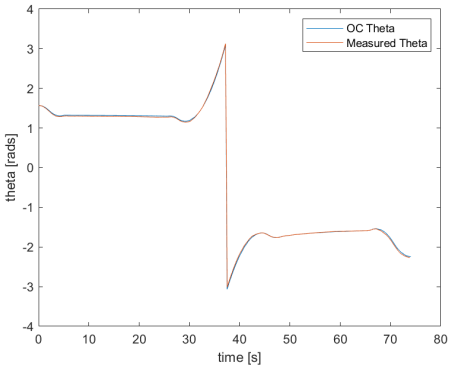
(d) Compared X Velocity



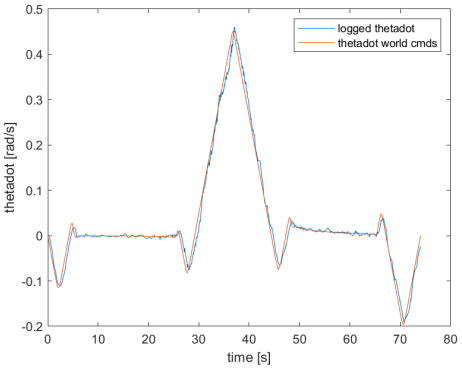
(e) Compared Y



(f) Compared Y Velocity

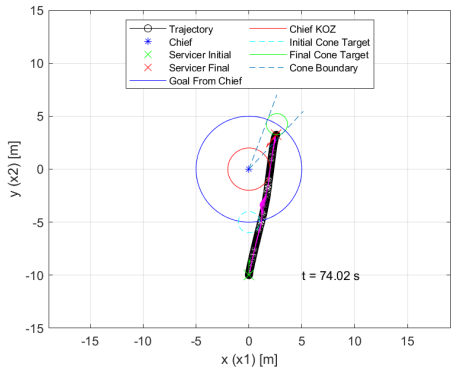


(g) Compared Theta

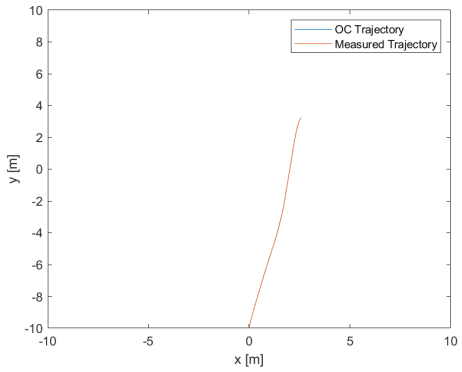


(h) Compared Theta Velocity

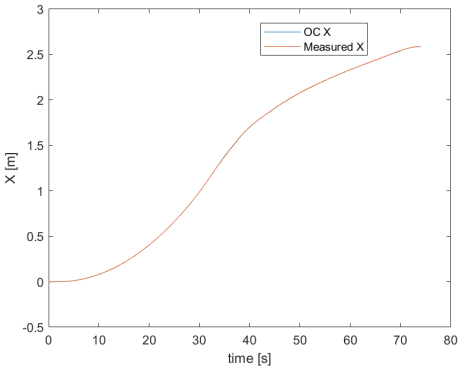
Figure D3: Test 4 Hardware Open Loop Control



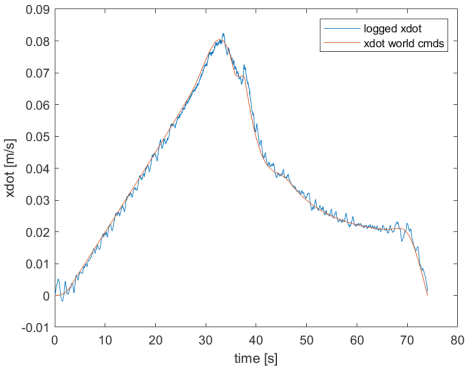
(a) Optimal Trajectory



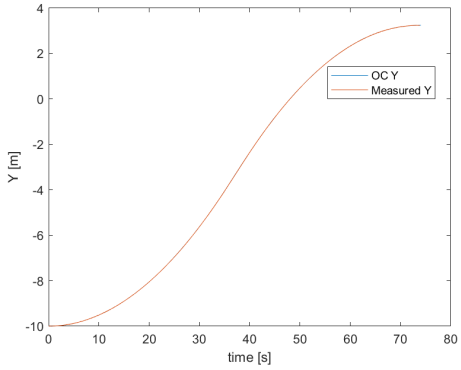
(b) Compared Trajectory



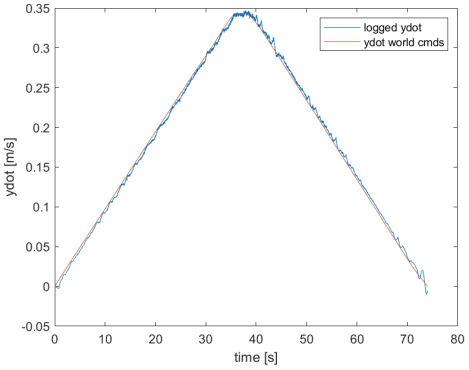
(c) Compared X



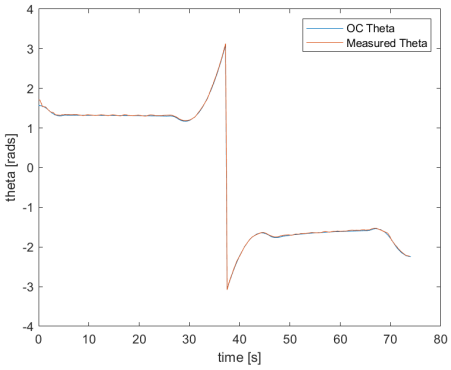
(d) Compared X Velocity



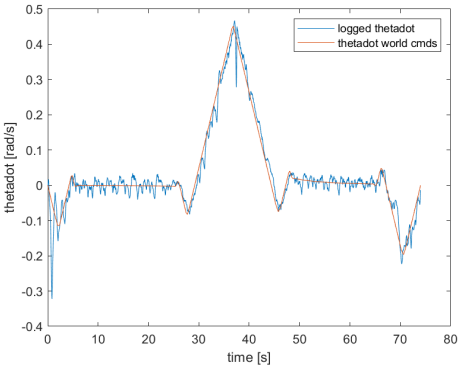
(e) Compared Y



(f) Compared Y Velocity

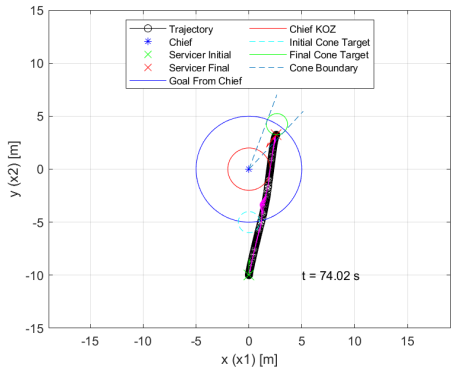


(g) Compared Theta

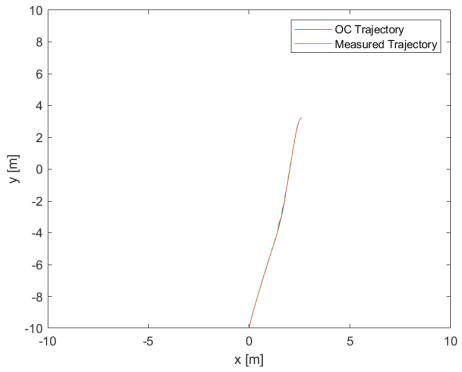


(h) Compared Theta Velocity

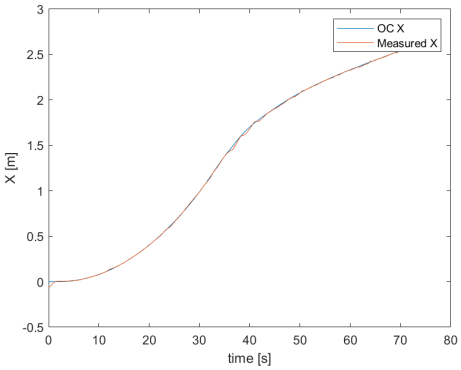
Figure D4: Test 4 Simulated Closed Loop Control



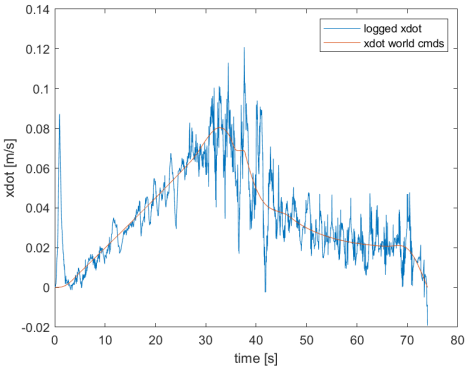
(a) Optimal Trajectory



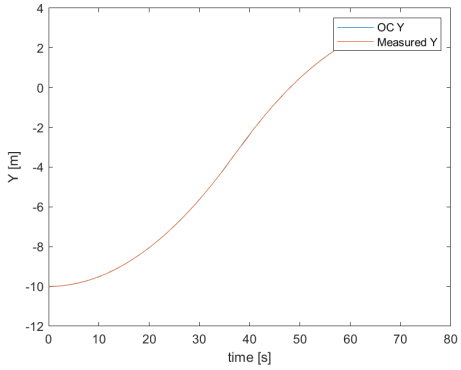
(b) Compared Trajectory



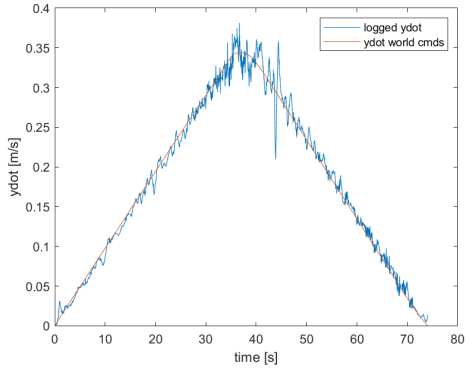
(c) Compared X



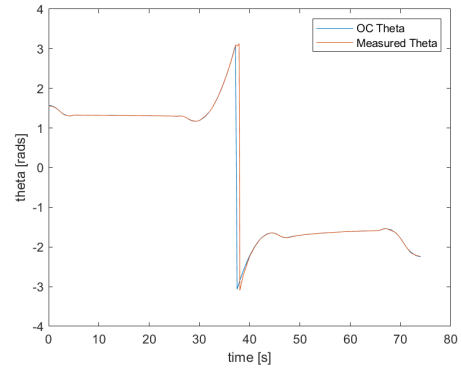
(d) Compared X Velocity



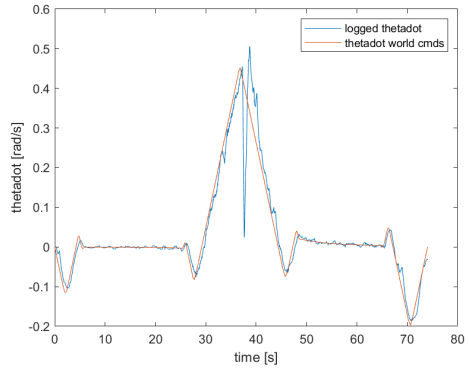
(e) Compared Y



(f) Compared Y Velocity



(g) Compared Theta

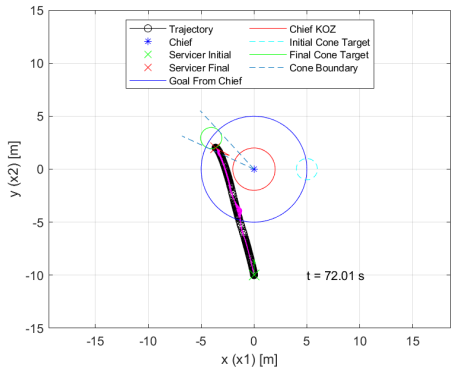


(h) Compared Theta Velocity

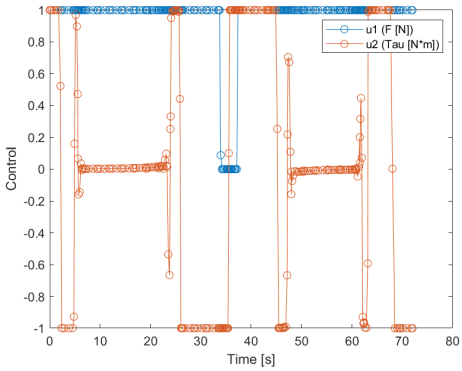
Figure D5: Test 4 Hardware Closed Loop Control

Appendix E. Test Case 5

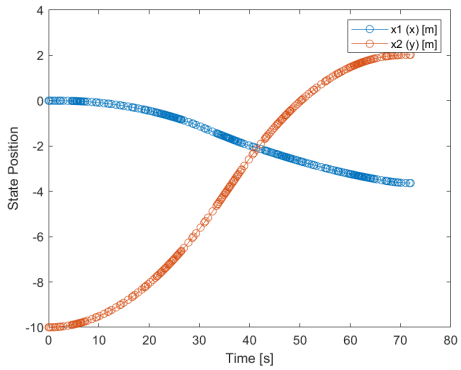
Appendix E includes all of the output figures for Test Case 5. Analysis of these results is found in Section 4.2.5. Test Case 5 uses Double Integrator dynamics and a rotating chief with a 2 deg/s rotation rate and an initial orientation of 0 radians.



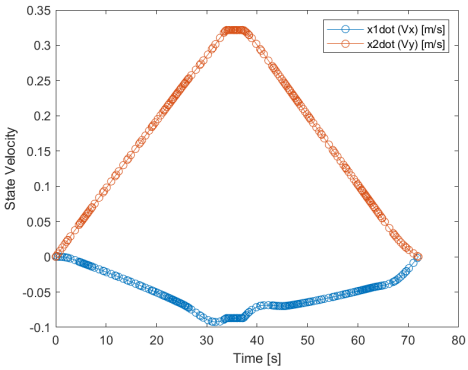
(a) Trajectory



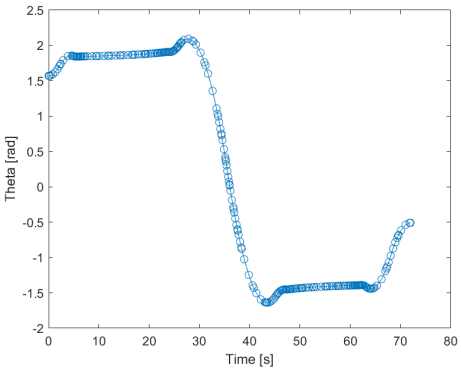
(b) Control History



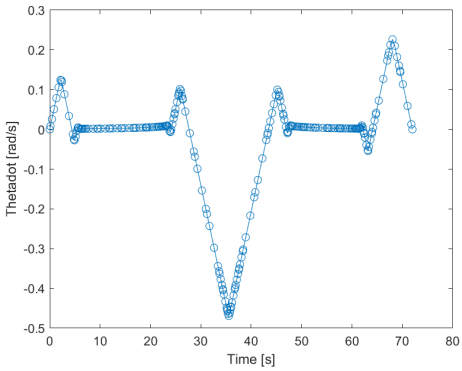
(c) Position



(d) Planar Velocities

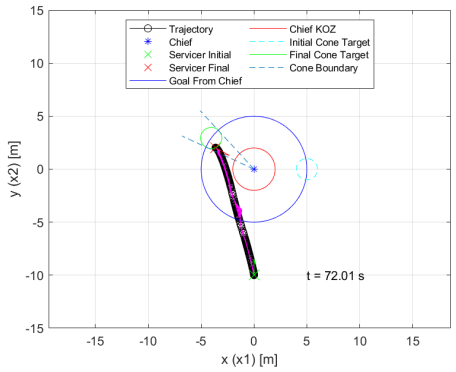


(e) Orientation

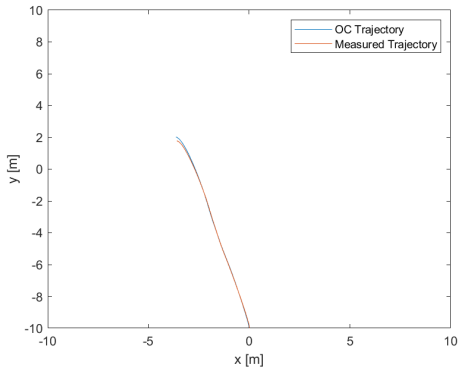


(f) Angular Velocity

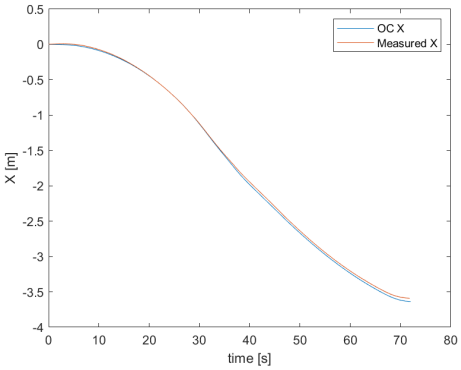
Figure E1: Test 5 Time Optimal Solution



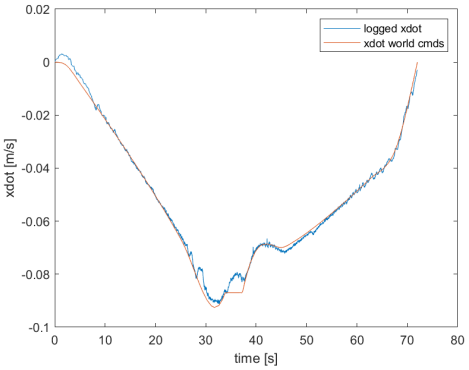
(a) Optimal Trajectory



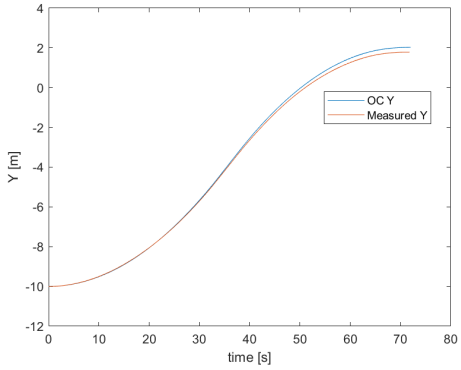
(b) Compared Trajectory



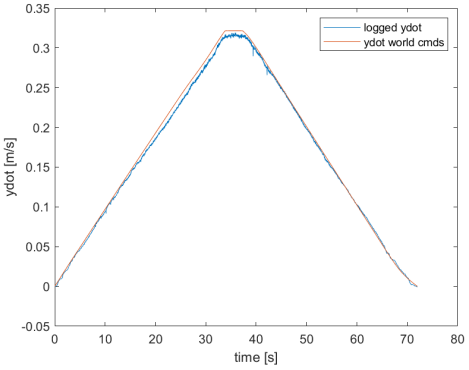
(c) Compared X



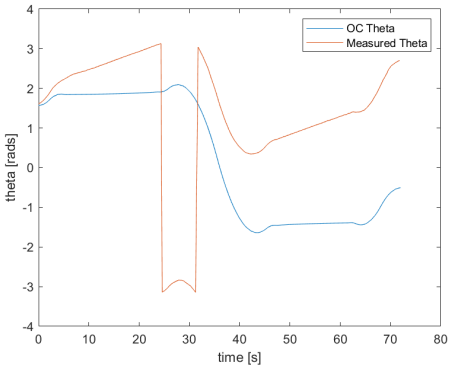
(d) Compared X Velocity



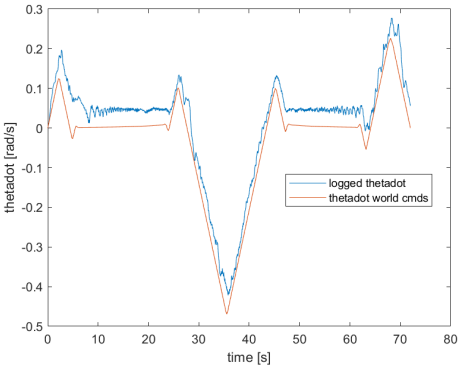
(e) Compared Y



(f) Compared Y Velocity

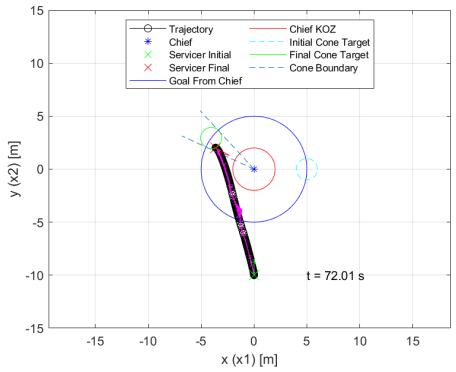


(g) Compared Theta

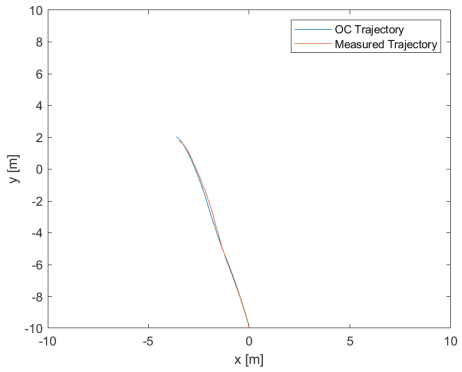


(h) Compared Theta Velocity

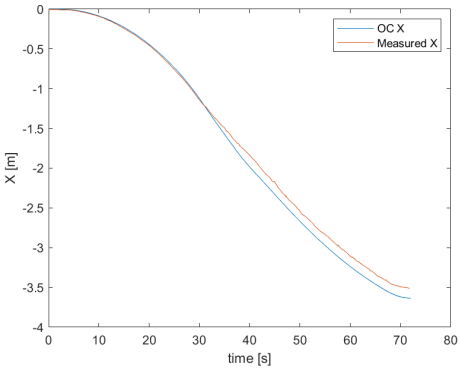
Figure E2: Test 5 Simulated Open Loop Control



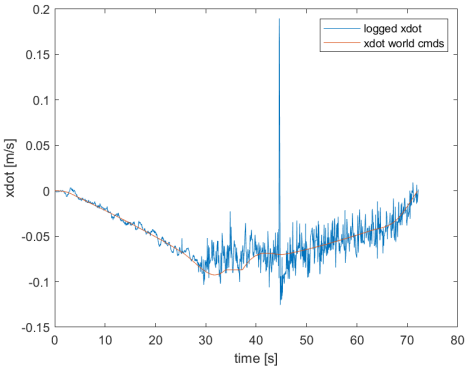
(a) Optimal Trajectory



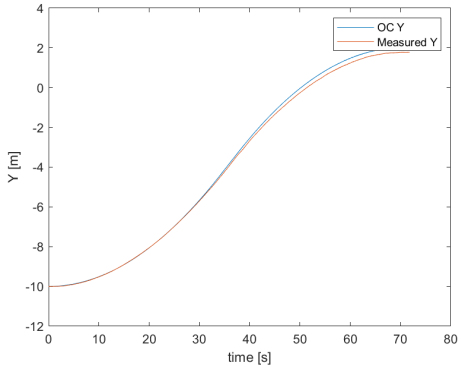
(b) Compared Trajectory



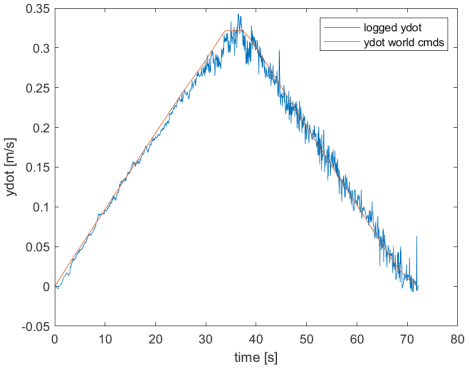
(c) Compared X



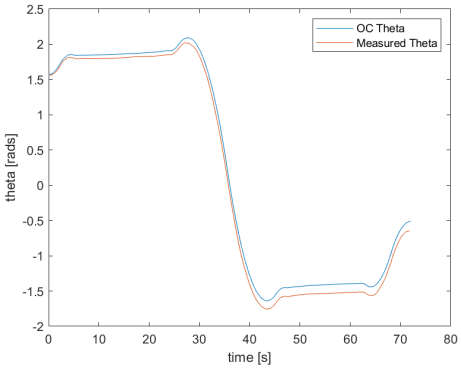
(d) Compared X Velocity



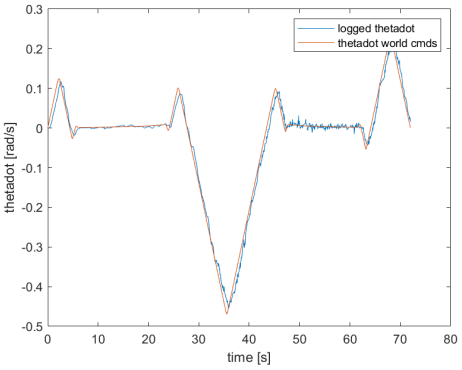
(e) Compared Y



(f) Compared Y Velocity

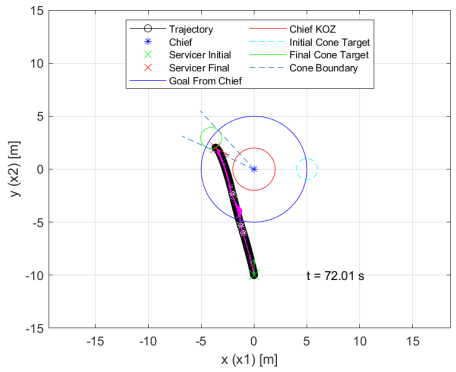


(g) Compared Theta

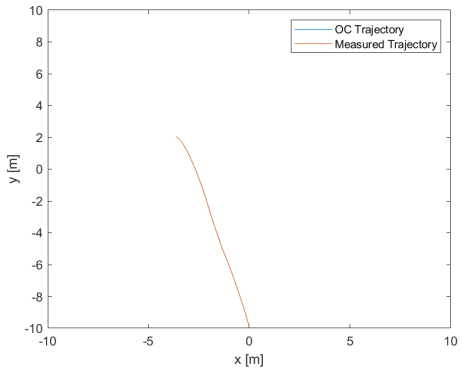


(h) Compared Theta Velocity

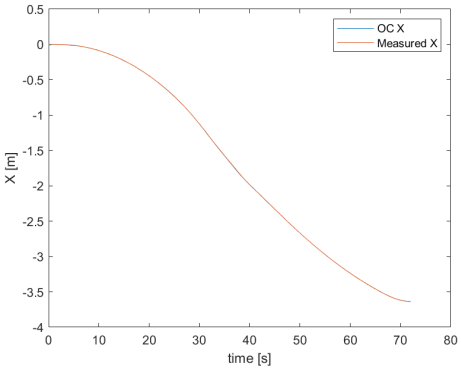
Figure E3: Test 5 Hardware Open Loop Control



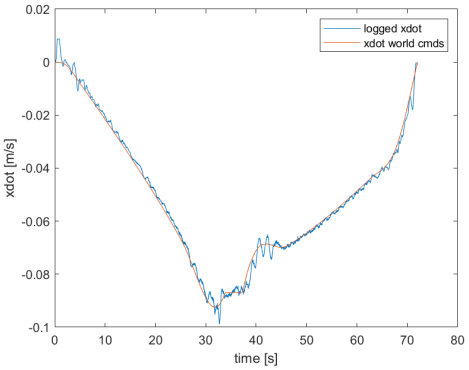
(a) Optimal Trajectory



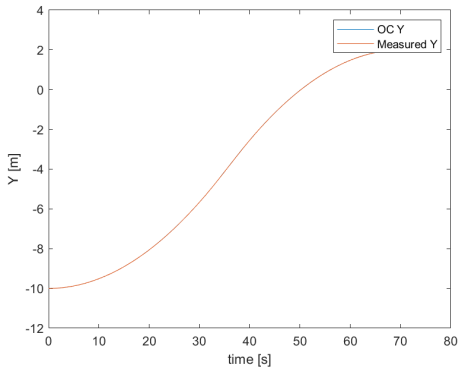
(b) Compared Trajectory



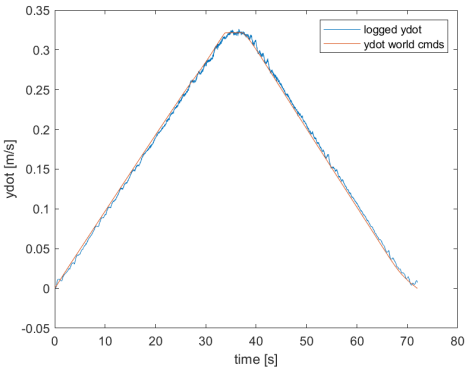
(c) Compared X



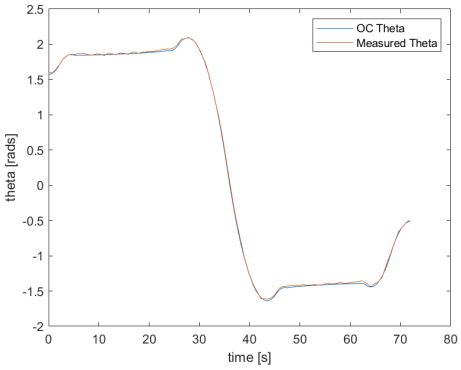
(d) Compared X Velocity



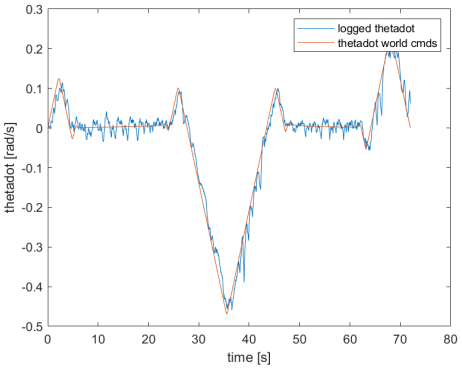
(e) Compared Y



(f) Compared Y Velocity

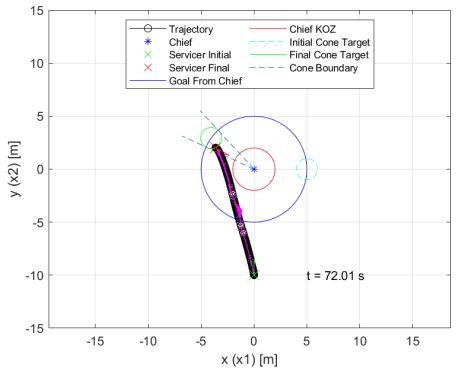


(g) Compared Theta

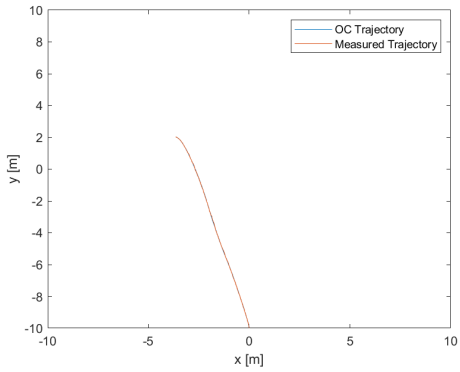


(h) Compared Theta Velocity

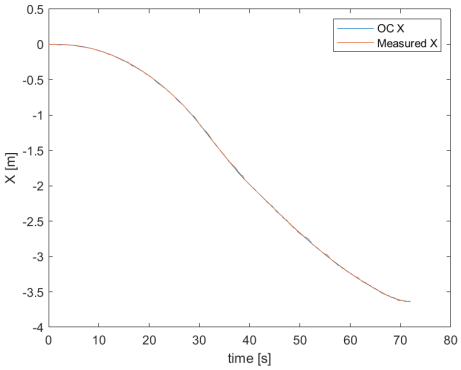
Figure E4: Test 5 Simulated Closed Loop Control



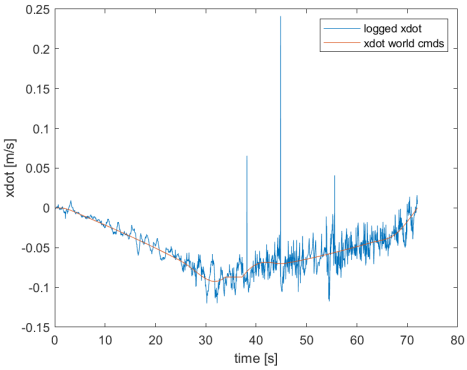
(a) Optimal Trajectory



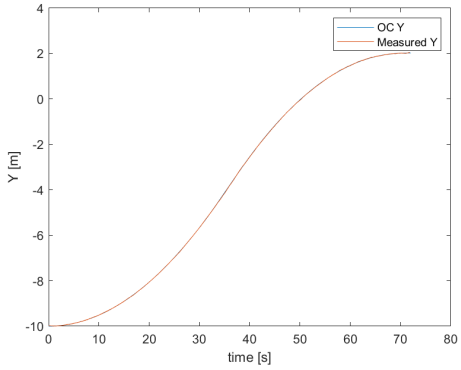
(b) Compared Trajectory



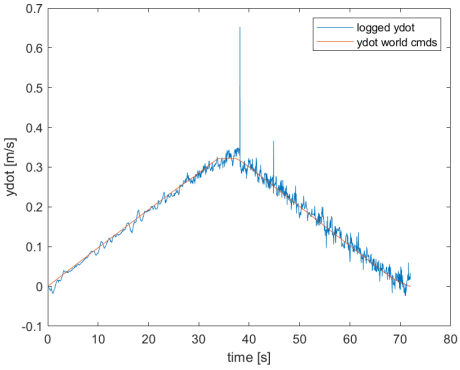
(c) Compared X



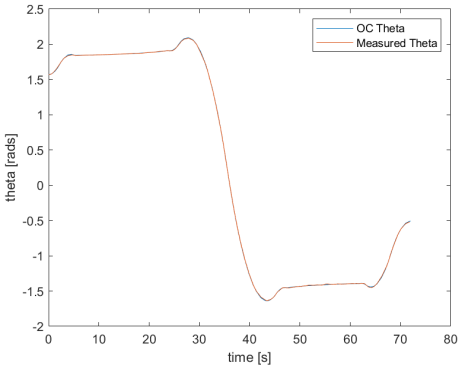
(d) Compared X Velocity



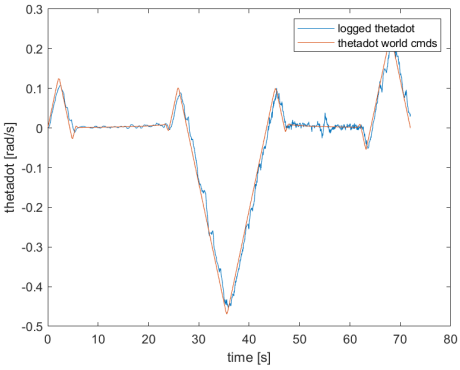
(e) Compared Y



(f) Compared Y Velocity



(g) Compared Theta

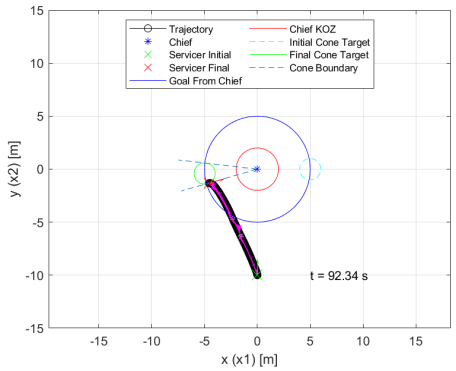


(h) Compared Theta Velocity

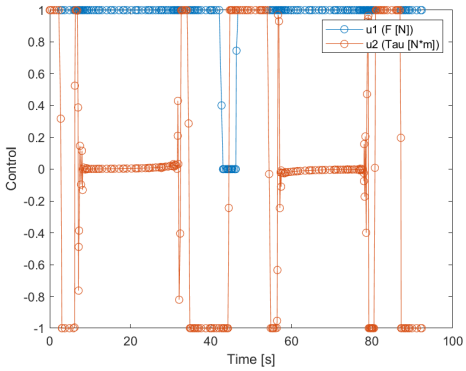
Figure E5: Test 5 Hardware Closed Loop Control

Appendix F. Test Case 6

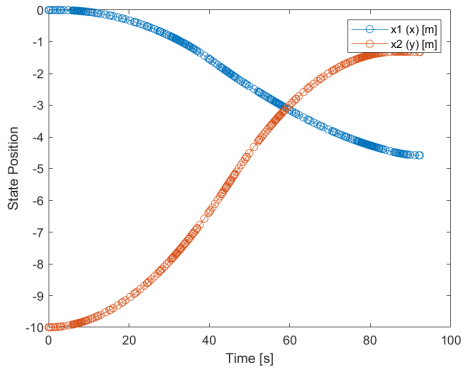
Appendix F includes all of the output figures for Test Case 6. Analysis of these results is found in Section 4.2.6. Test Case 6 uses Clohessy-Wiltshire dynamics and a rotating chief with a 2 deg/s rotation rate and an initial orientation of 0 radians. Test Case 6 also uses a deputy with a mass of 200 kg instead of 100 kg.



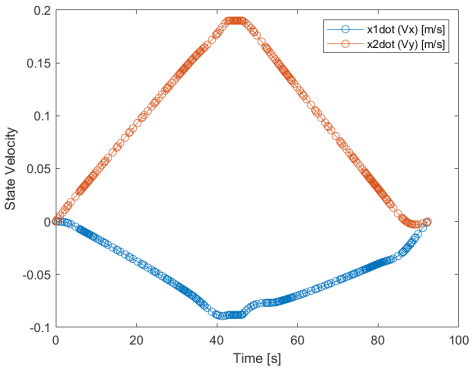
(a) Trajectory



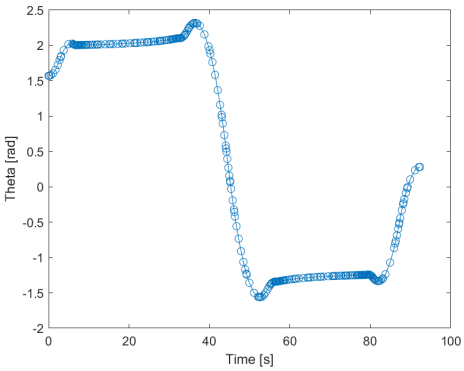
(b) Control History



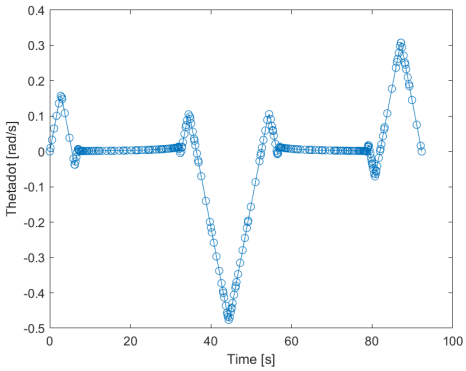
(c) Position



(d) Planar Velocities

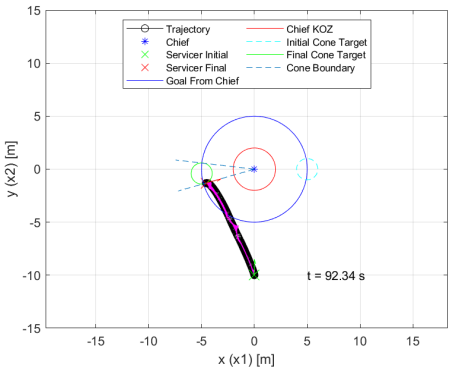


(e) Orientation

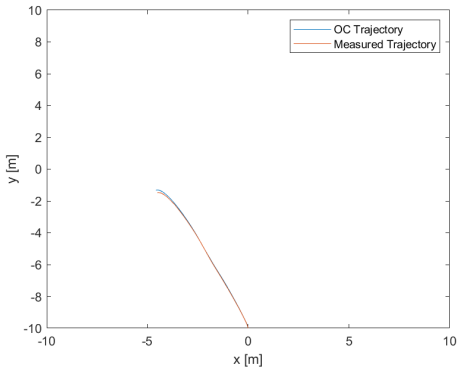


(f) Angular Velocity

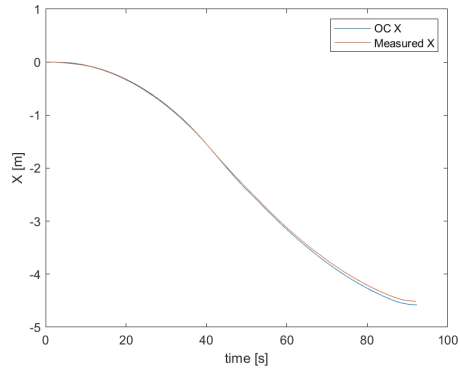
Figure F1: Test 6 Time Optimal Solution



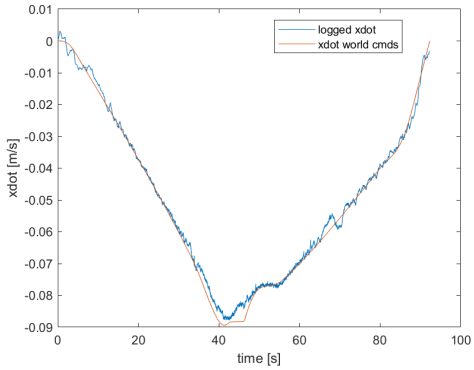
(a) Optimal Trajectory



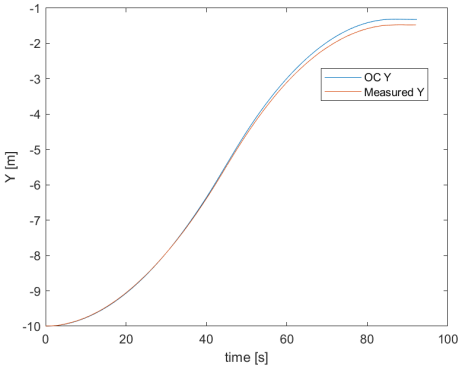
(b) Compared Trajectory



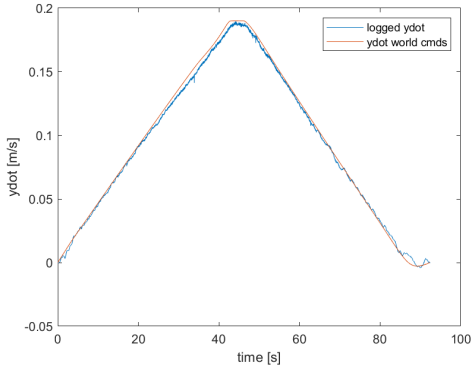
(c) Compared X



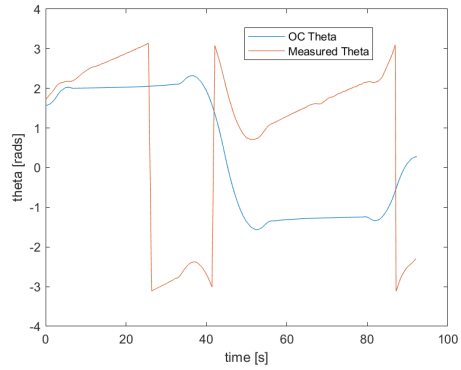
(d) Compared X Velocity



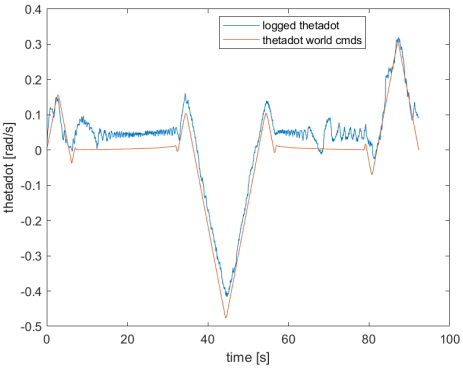
(e) Compared Y



(f) Compared Y Velocity

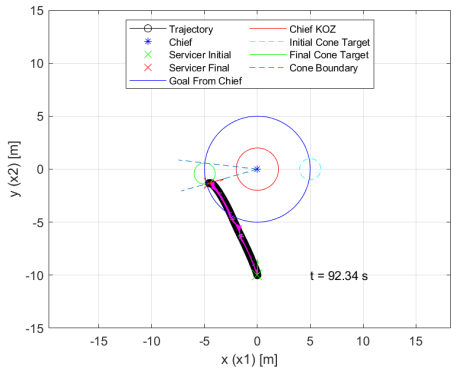


(g) Compared Theta

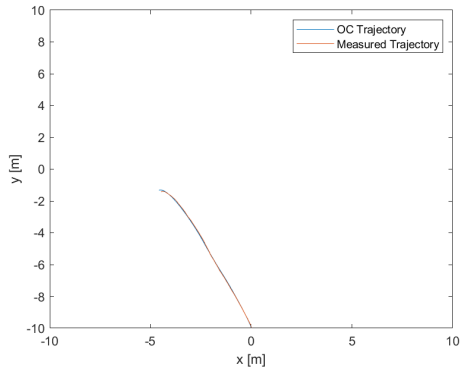


(h) Compared Theta Velocity

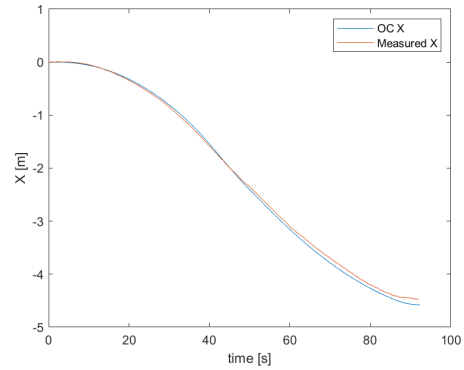
Figure F2: Test 6 Simulated Open Loop Control



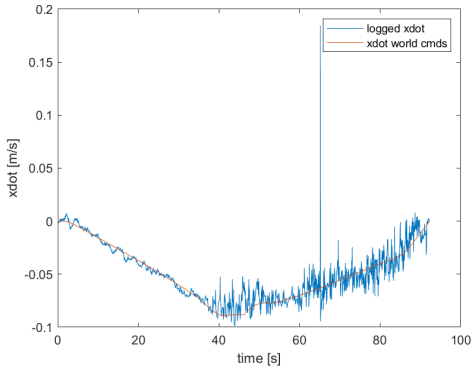
(a) Optimal Trajectory



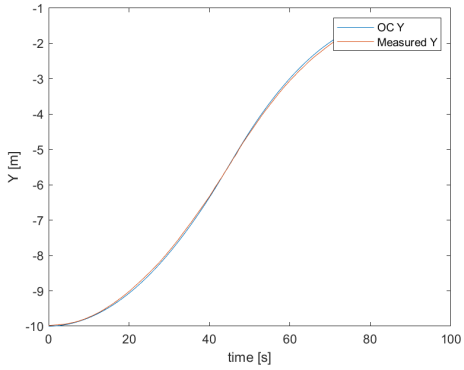
(b) Compared Trajectory



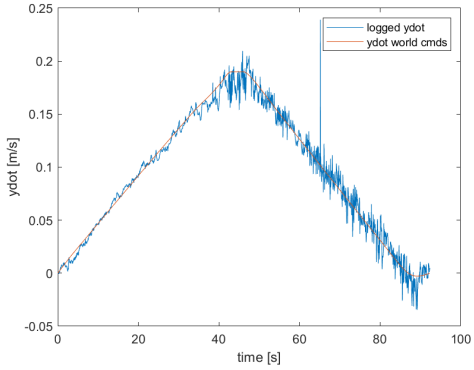
(c) Compared X



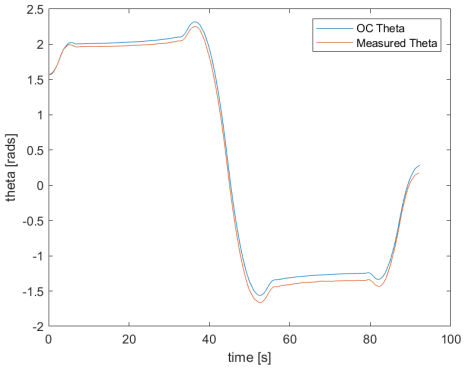
(d) Compared X Velocity



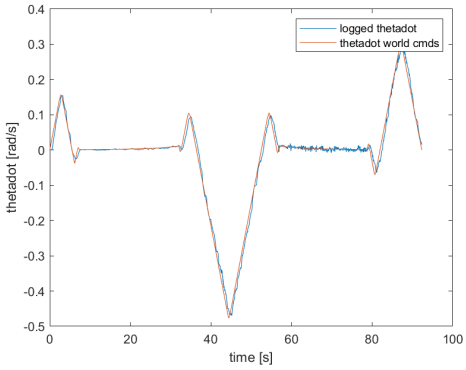
(e) Compared Y



(f) Compared Y Velocity

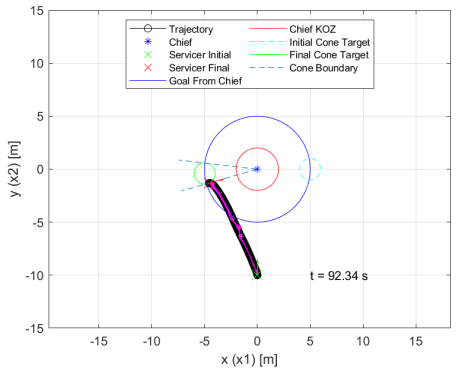


(g) Compared Theta

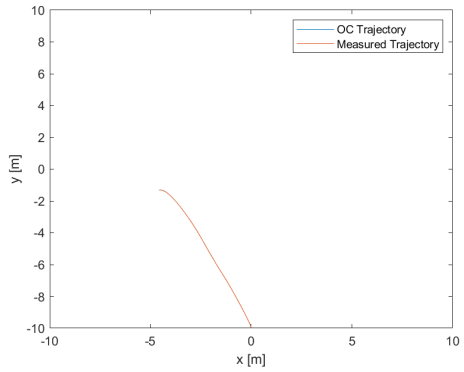


(h) Compared Theta Velocity

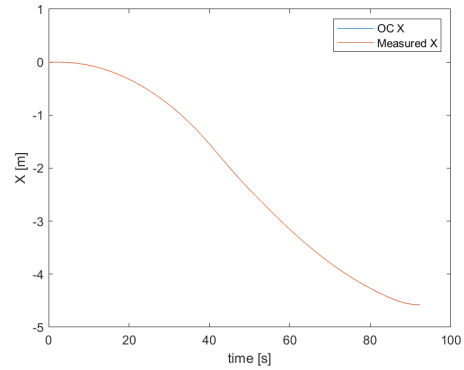
Figure F3: Test 6 Hardware Open Loop Control



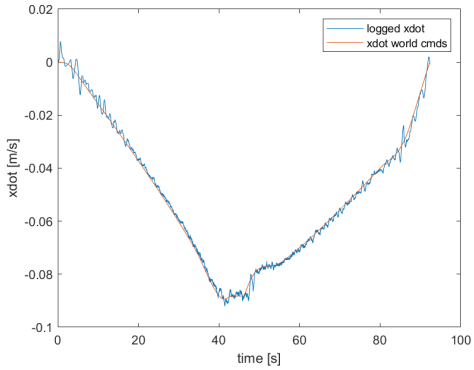
(a) Optimal Trajectory



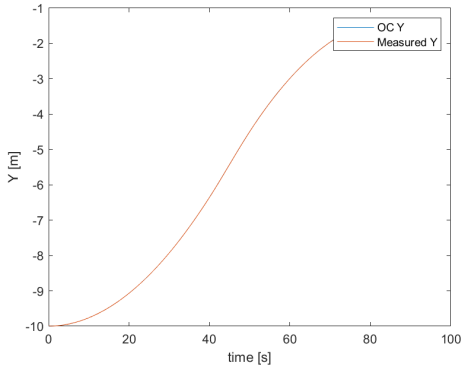
(b) Compared Trajectory



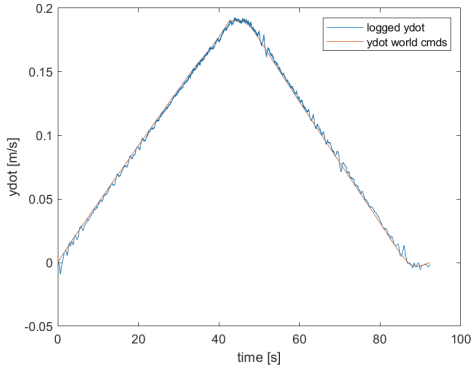
(c) Compared X



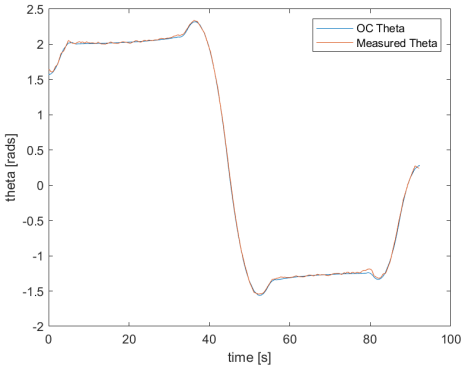
(d) Compared X Velocity



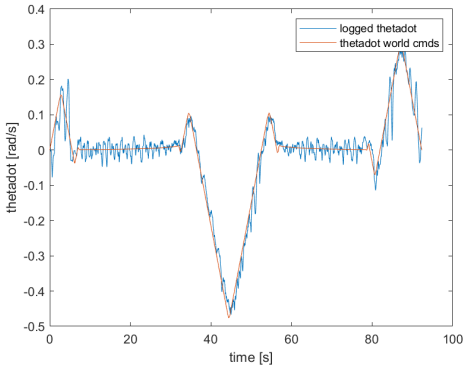
(e) Compared Y



(f) Compared Y Velocity

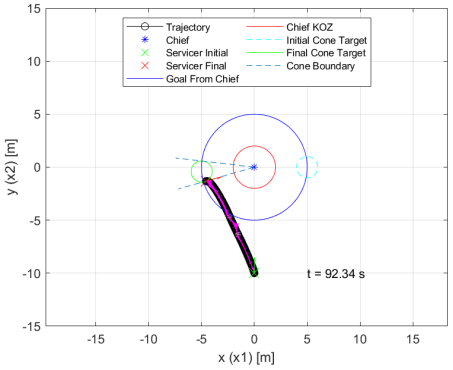


(g) Compared Theta

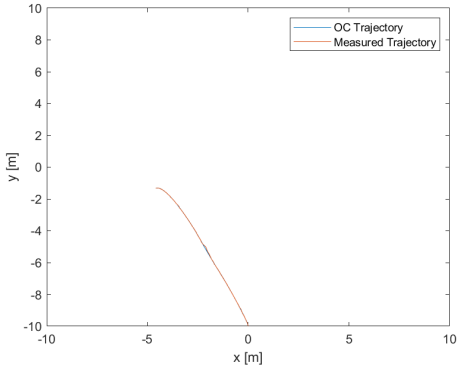


(h) Compared Theta Velocity

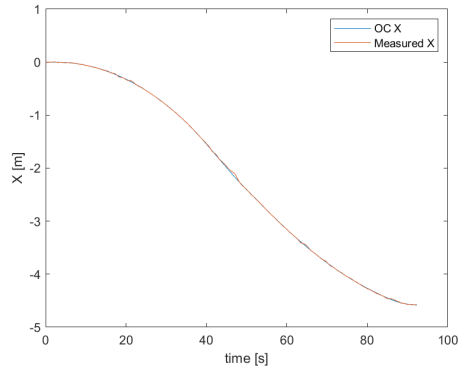
Figure F4: Test 6 Simulated Closed Loop Control



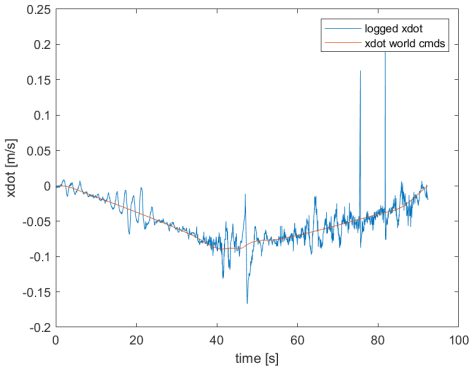
(a) Optimal Trajectory



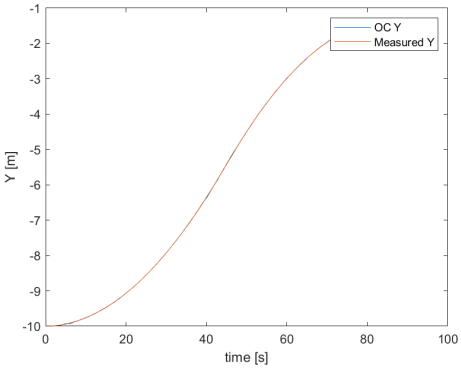
(b) Compared Trajectory



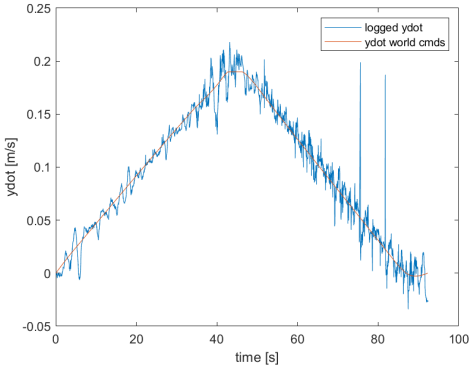
(c) Compared X



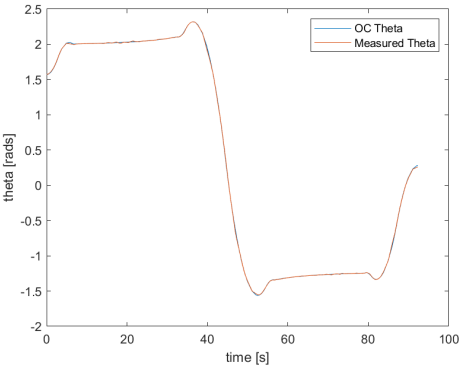
(d) Compared X Velocity



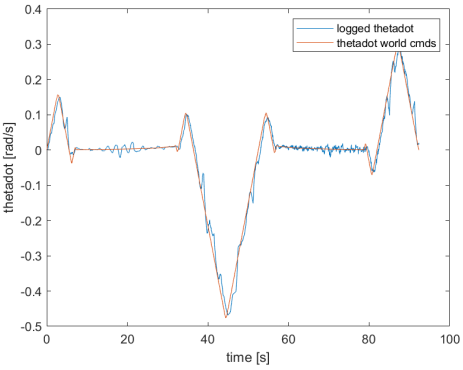
(e) Compared Y



(f) Compared Y Velocity



(g) Compared Theta

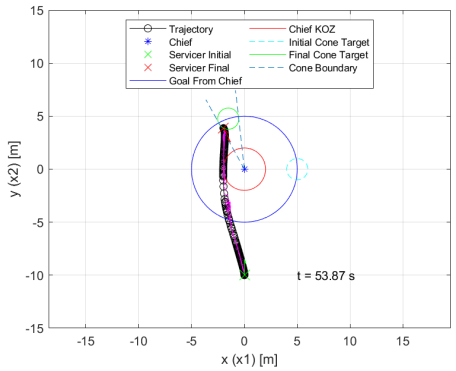


(h) Compared Theta Velocity

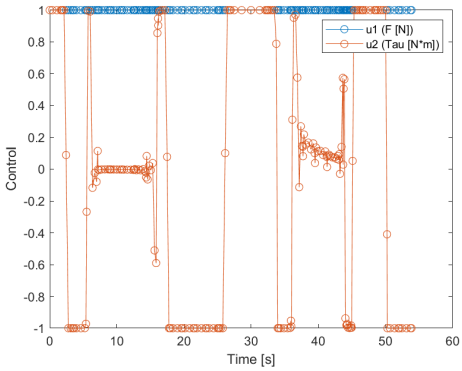
Figure F5: Test 6 Hardware Closed Loop Control

Appendix G. Test Case 7

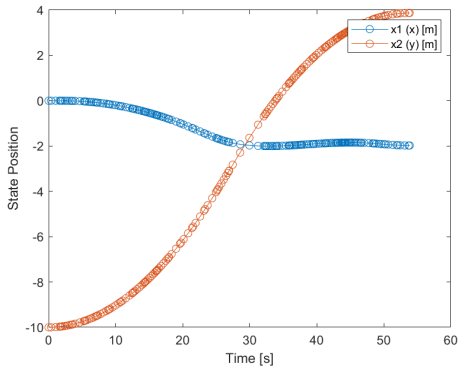
Appendix G includes all of the output figures for Test Case 7. Analysis of these results is found in Section 4.2.7. Test Case 7 uses Clohessy-Wiltshire dynamics and a rotating chief with a 2 deg/s rotation rate and an initial orientation of 0 radians. Test Case 7 also uses a deputy with a mass of 50 kg instead of 100 kg.



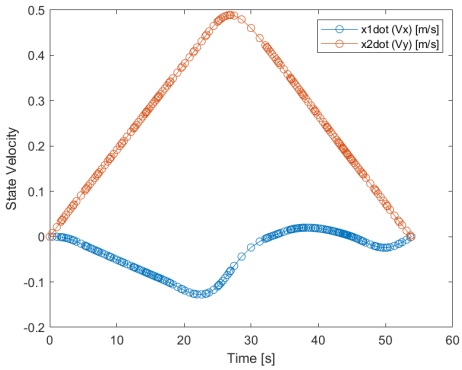
(a) Trajectory



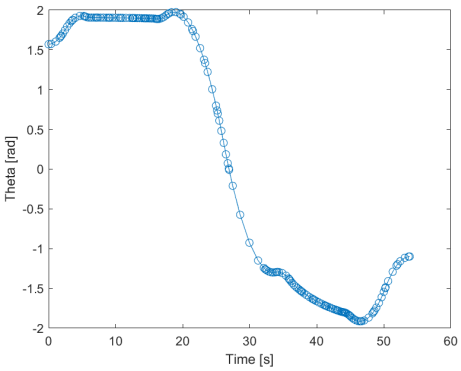
(b) Control History



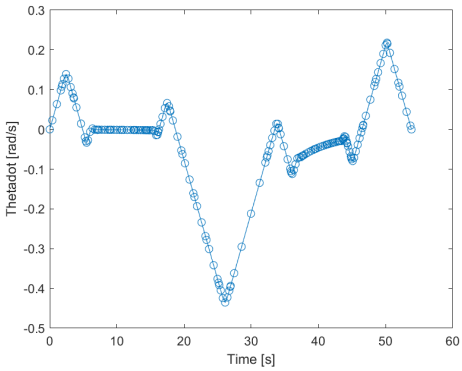
(c) Position



(d) Planar Velocities

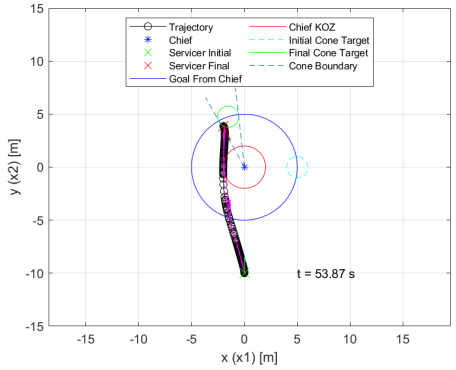


(e) Orientation

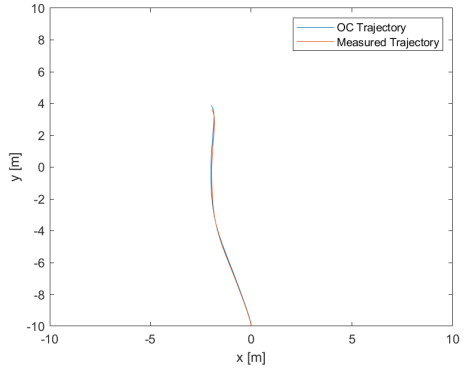


(f) Angular Velocity

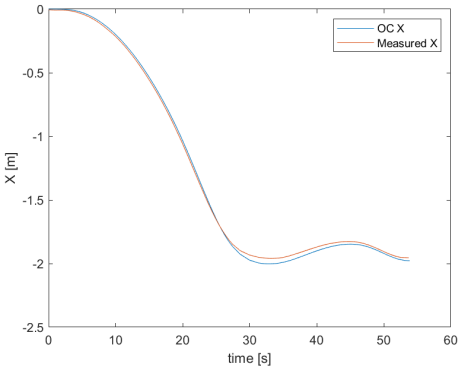
Figure G1: Test 7 Time Optimal Solution



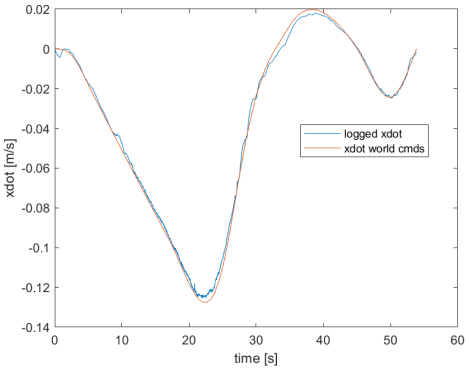
(a) Optimal Trajectory



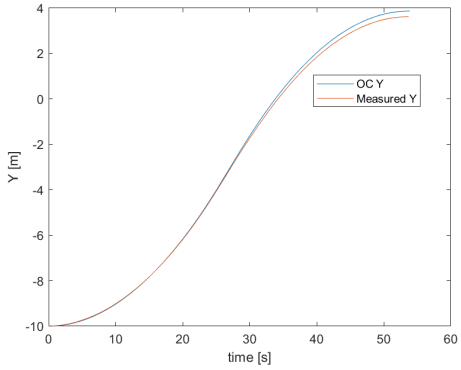
(b) Compared Trajectory



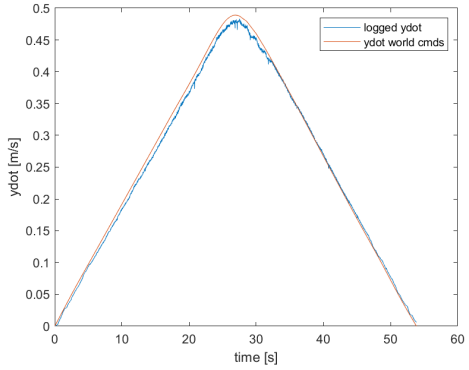
(c) Compared X



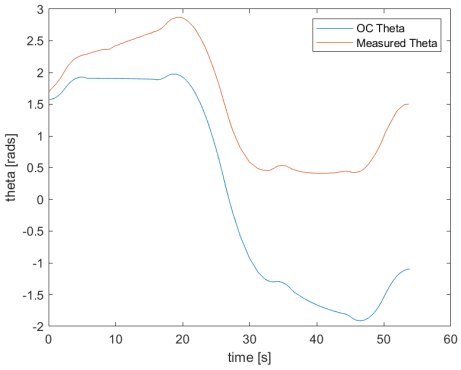
(d) Compared X Velocity



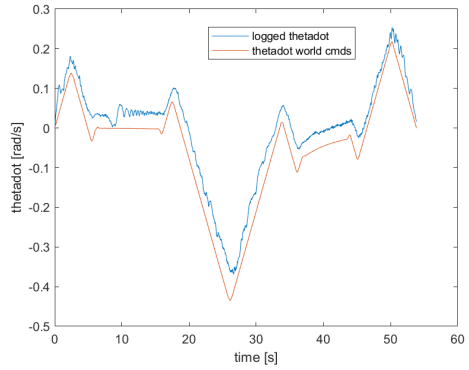
(e) Compared Y



(f) Compared Y Velocity

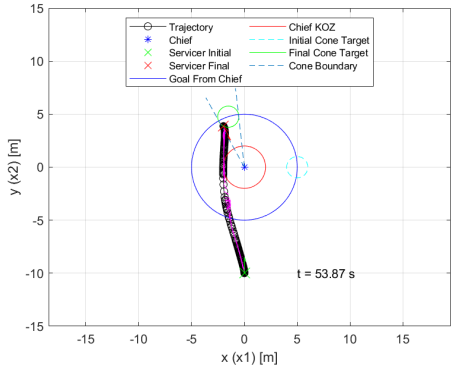


(g) Compared Theta

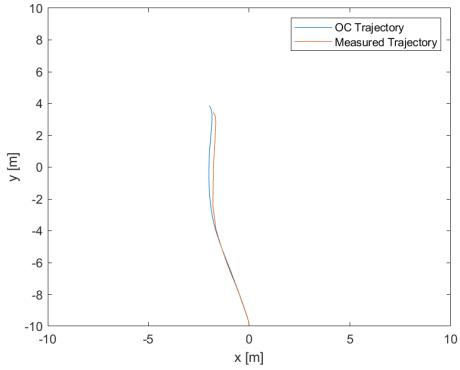


(h) Compared Theta Velocity

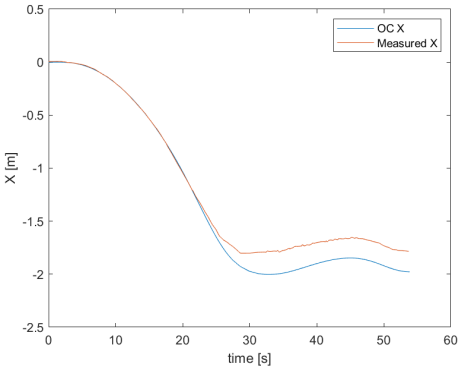
Figure G2: Test 7 Simulated Open Loop Control



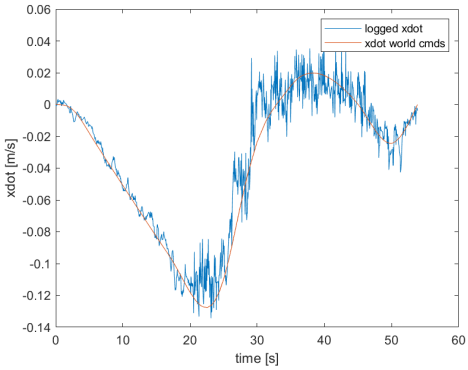
(a) Optimal Trajectory



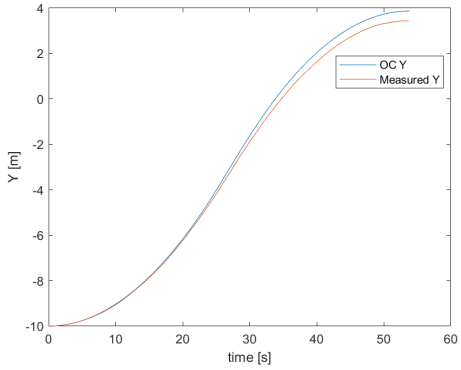
(b) Compared Trajectory



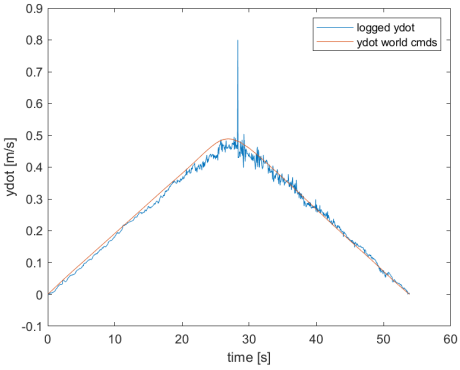
(c) Compared X



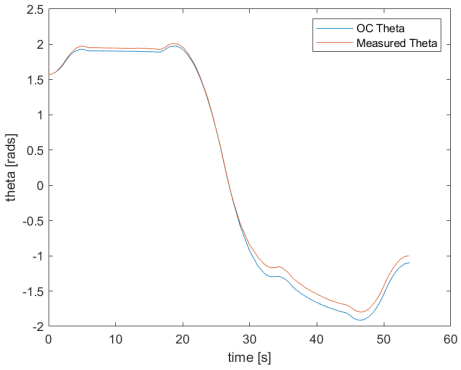
(d) Compared X Velocity



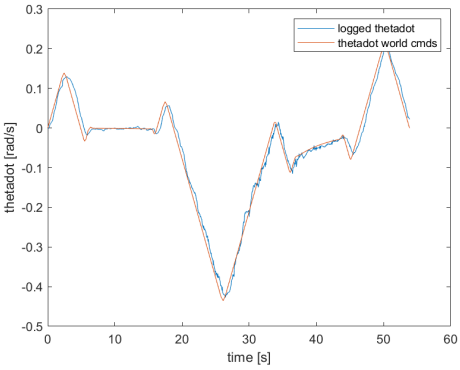
(e) Compared Y



(f) Compared Y Velocity

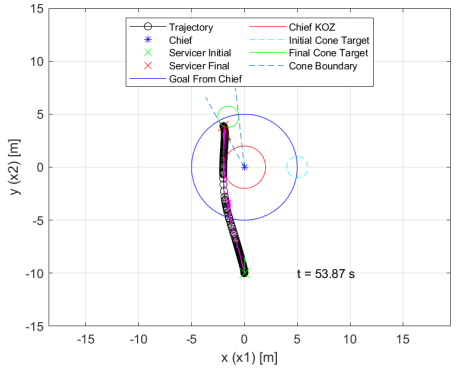


(g) Compared Theta

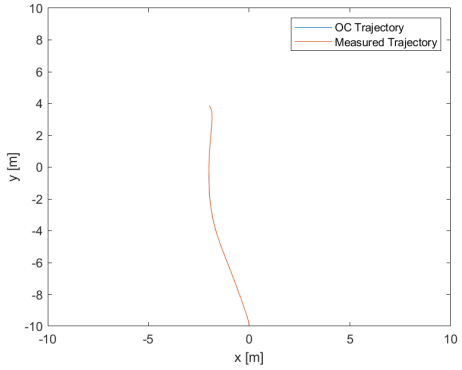


(h) Compared Theta Velocity

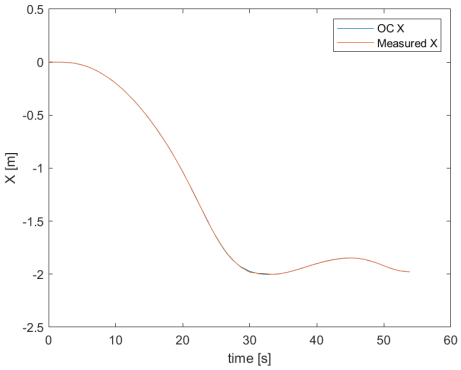
Figure G3: Test 7 Hardware Open Loop Control



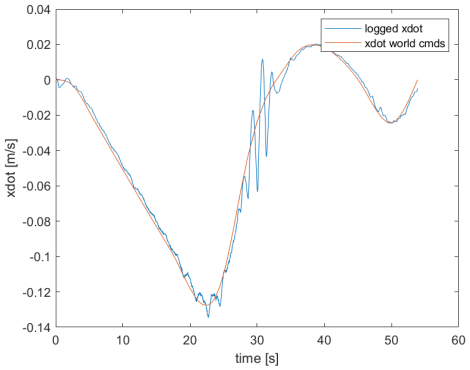
(a) Optimal Trajectory



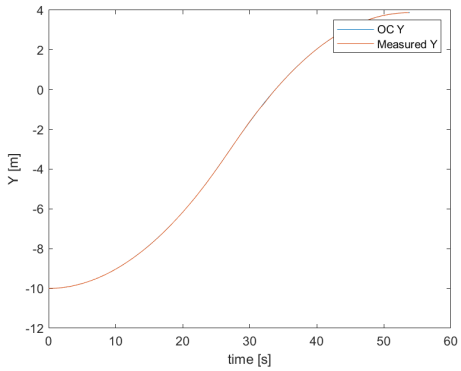
(b) Compared Trajectory



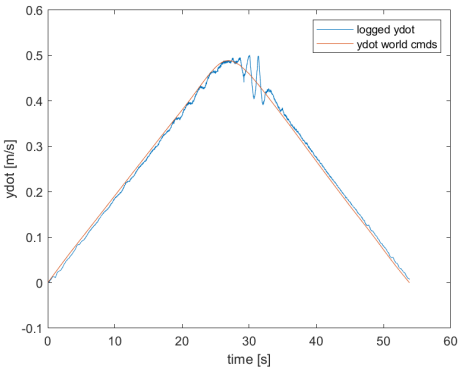
(c) Compared X



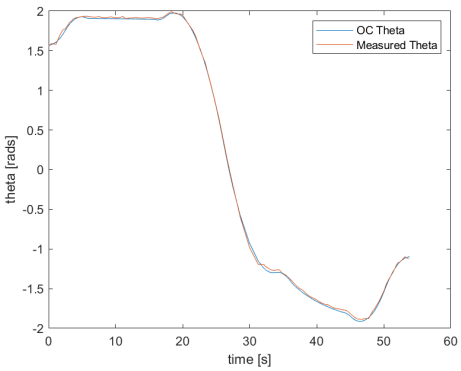
(d) Compared X Velocity



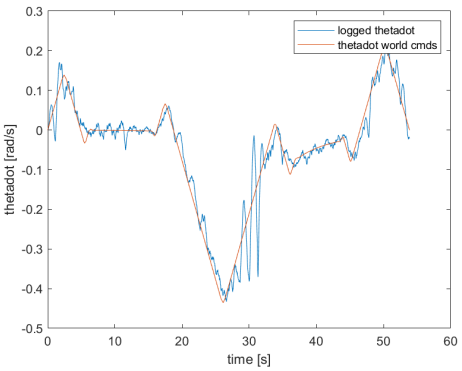
(e) Compared Y



(f) Compared Y Velocity

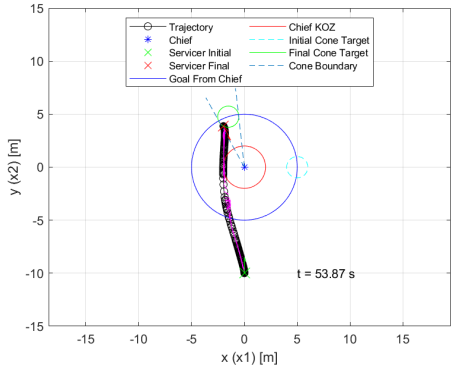


(g) Compared Theta

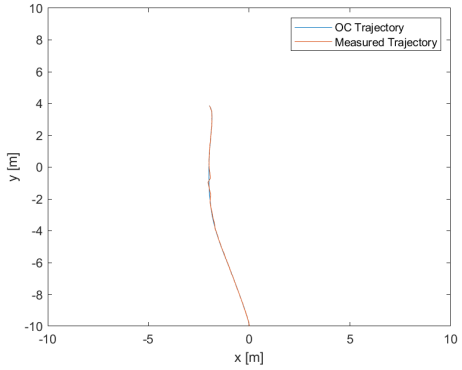


(h) Compared Theta Velocity

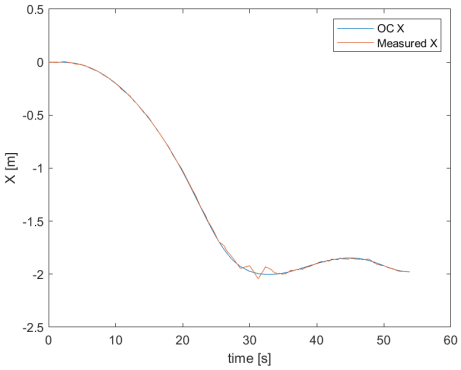
Figure G4: Test 7 Simulated Closed Loop Control



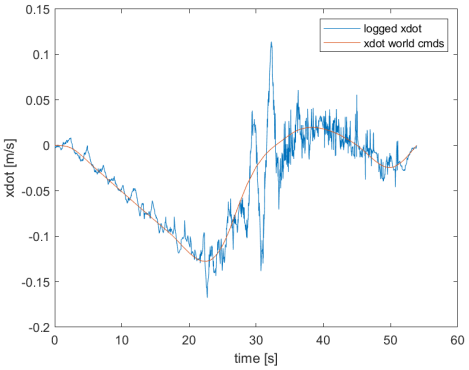
(a) Optimal Trajectory



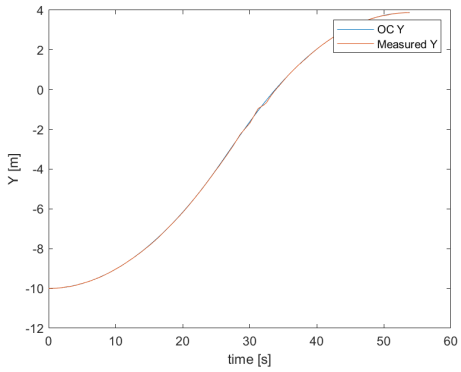
(b) Compared Trajectory



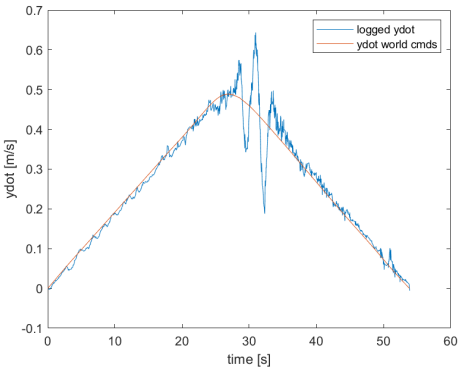
(c) Compared X



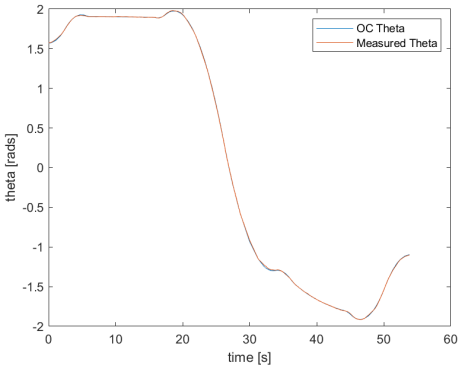
(d) Compared X Velocity



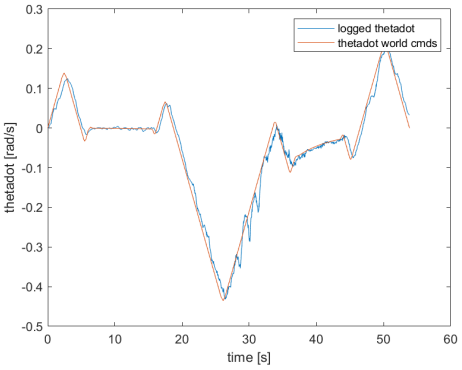
(e) Compared Y



(f) Compared Y Velocity



(g) Compared Theta



(h) Compared Theta Velocity

Figure G5: Test 7 Hardware Closed Loop Control

Bibliography

1. On-orbit satellite servicing study: Project report, 2010.

2. David C. Woffinden and David K. Geller. Navigating the road to autonomous orbital rendezvous. *Journal of Spacecraft and Rockets*, 44, 2007.

3. Richard Zappulla, Josep Virgili-Llop, Costantinos Zagaris, Hyeongjun Park, Alanna Sharp, and Marcello Romano. Floating spacecraft simulator test bed for the experimental testing of autonomous guidance, navigation, and control of spacecraft proximity maneuvers and operations. 2016.

4. Richard Zappulla, Josep Virgili-Llop, Costantinos Zagaris, Hyeongjun Park, and Marcello Romano. Dynamic air-bearing hardware-in-the-loop testbed to experimentally evaluate autonomous spacecraft proximity maneuvers. volume 54, 2017.

5. Hyeongjun Park, Richard Zappulla, Costantinos Zagaris, Josep Virgili-Llop, and Marcello Romano. Nonlinear model predictive control for spacecraft rendezvous and docking with a rotating target. volume 160, 2017.

6. Joshua Davis, John Mayberry, and Jay Penn. On-orbit servicing: Inspection, repair, refuel, upgrade, and assembly of satellites in space, 2019.

7. Roberto Opromolla, Giancarmine Fasano, Giancarlo Rufino, and Michele Grassi. A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations. *Progress in Aerospace Sciences*, 93, 2017.

8. W. H. CLOHESSY and R. S. WILTSHIRE. Terminal guidance system for satellite rendezvous. *Journal of the Aerospace Sciences*, 27, 1960.

9. Michael A. Patterson and Anil V. Rao. Gpops - ii: A matlab software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature

collocation methods and sparse nonlinear programming. *ACM Transactions on Mathematical Software*, 41, 2014.

10. Daniel P. Scharf, Fred Y. Hadaegh, and Bryan H. Kang. A survey of spacecraft formation flying guidance. *Proceedings of the International Symposium Formation Flying*, 2002.

11. Kyle T. Alfriend, Srinivas R. Vadali, Pini Gurfil, Jonathan P. How, and Louis S. Breger. *Spacecraft formation flying: Dynamics, control and navigation.* 2009.

12. Daniel P. Scharf, Fred Y. Hadaegh, and Bryan H. Kang. On the validity of the double integrator approximation in deep space formation flying. 2002.

13. Roland Siegwart, Illah R Nourbakhsh, and Davide Scaramuzza. *Introduction to Autonomous Mobile Robots, Second Edition*, volume 23. 2011.

14. José Gonçalves, José Lima, and Paulo Costa. Real time tracking of an omnidirectional robot - an extended kalman filter approach. volume 2 RA, 2008.

15. Tamás Kalmár-Nagy, Raffaello D'Andrea, and Pritam Ganguly. Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle. *Robotics and Autonomous Systems*, 46, 2004.

16. Donald E. Kirk. *Optimal Control Theory: An Introduction.* Prentice-Hall Inc., 1970.

17. Joseph A. Starek, Behçet Açıkmeşe, Issa A. Nesnas, and Marco Pavone. Spacecraft autonomy challenges for next-generation space missions, 2016.

18. Christopher Jewison, R. Scott Erwin, and Alvar Saenz-Otero. Model predictive control with ellipsoid obstacle constraints for spacecraft rendezvous. volume 28, 2015.

19. Morgan Quigley, Brian Gerkey, and William D. Smart. *Programming Robots with ROS A Practical Introduction to the Robot Operating System*, volume 53. 2015.

20. Beard R and McLain T. *Introduction to Feedback Control using Design Studies*. 2020.

Acronyms

2-DOF two degrees of freedom. 11

3-DOF three degrees of freedom. 5, 6, 7, 12, 13, 14, 28, 32, 46, 47, 53, 105, 106

ANT Autonomy and Navigation Technology. 4, 56, 93, 103

APF Artificial Potential Functions. 107

COE Classical Orbital Element. 8

CW Clohessy-Wiltshire. 3, 4, 10, 11, 12, 28, 46, 47, 48, 57, 65, 77, 79, 81, 83, 85

DDOF differential degrees of freedom. 19

DHCP Dynamic Host Configuration Protocol. 42

DI double integrator. 11, 77, 79, 81, 83

DOF degrees of freedom. 3, 4, 11, 12, 20, 25, 46

ECI Earth-Centered-Inertial. 7, 8, 9

GEO Geosynchronous Equatorial Orbit. 1, 4, 8, 9, 10, 11

GNC Guidance, Navigation, and Control. 2, 3

GUI graphical user interface. 27

IDE Integrated Development Environment. 36

KOZ keep out zone. 54

LEO Low Earth Orbit. 1

LiPo lithium-ion polymer. 31, 34

LQR linear-quadratic regulator. 50

LVLH Local Vertical, Local Horizontal. 4, 9, 10, 16, 46, 47, 54, 56, 107

MPC Model Predictive Control. 107

NLP nonlinear programming. 25

OS Operating System. 34

PID proportional-integral-derivative. 49

RMSE root-mean-square error. 63, 65, 69, 72, 101, 102, 103, 106

ROS Robot Operating System. 4, 5, 6, 26, 27, 28, 30, 34, 35, 36, 37, 42, 43, 44, 45, 46, 51, 53, 105, 106

SSH Secure Shell. 35, 43, 45

UAV unmanned aerial vehicle. 105

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704–0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
25–03–2021	Master's Thesis	Sept 2019 — Mar 2021

4. TITLE AND SUBTITLE

LOW-COST TERRESTRIAL DEMONSTRATION OF AUTONOMOUS SATELLITE PROXIMITY OPERATIONS

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Hewitt, Zackary, 1st Lt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-21-M-047

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory Sensors Directorate (AFRL/RY)
2241 Avionics Cir
WPAFB OH 45433-7765
COMM: 937-713-8116
Email: trevor.bihl.2@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/RYAR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The lack of satellite servicing capabilities significantly impacts the development and operation of current orbital assets. With autonomous solutions under consideration for servicing, the purpose of this research is to build and validate a low-cost hardware platform to expedite the development of autonomous satellite proximity operations. This research aims to bridge the gap between simulation and existing higher fidelity hardware testing with an affordable alternative. An omnidirectional variant of the commercially available TurtleBot3 mobile robot is presented as a 3-DOF testbed that demonstrates a satellite servicing inspection scenario. Reference trajectories for the scenario are generated via optimal control using the commercial solver GPOPS-II, and results from simulation and hardware demonstration are presented. Recommendations are then given for using the platform as a rapid method for experimentally verifying various satellite control algorithms.

15. SUBJECT TERMS
autonomous satellite servicing, Clohessy-Wiltshire (CW) dynamics, hardware demonstration, holonomic system, motion capture, omnidirectional mobile robot, optimal control, relative orbital motion, Robot Operating System (ROS), simulation, three degrees of freedom (3-DOF) testbed

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Robert Leishman, AFIT/ENG
U	U	U	UU	168	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636 x4755; robert.leishman@afit.edu