STUDY OF AUTONOMOUS CAPTURE AND DETUMBLE OF NON-COOPERATIVE TARGET BY A FREE-FLYING SPACE MANIPULATOR USING AN AIR-BEARING PLATFORM

Lucas Santaguida

A thesis submitted to the Faculty of Graduate Studies

in partial fulfillment of the requirements for the degree of

Master of Applied Science

Graduate Program in Mechanical Engineering

York University

Toronto, Ontario

June 2020

Abstract

This thesis developed a 3-DOF satellite simulator with an attached 3-DOF manipulator to capture and detumble a target satellite simulator. Existing systems are heavily dependent on external systems to compute the position and orientation of the chaser and target satellite simulators. Using external sensors and high-power computers allows their systems to have high accuracy and high sampling frequencies. This approach is not reflective of the challenges faced by an on-orbit servicing spacecraft as all positioning of the space vehicle is computed on-board. In addition, their systems use the same external sensors to determine the position and orientation of the target simulator and transmit it to the chaser. A true on-orbit servicing vehicle would need to sense and compute the target simulators position and orientation relative to itself. The simulator developed in this thesis addresses these issues by computing its own position using a star-tracking system and computes the relative position and orientation of the target simulator using a monocular camera. The simulator was developed to act as a testbed for on-orbit servicing technologies. Different sensors, path planning and control algorithms can be implemented to test their effectiveness before implementation on a servicing vehicle. To demonstrate this, a PD

controller as well as an adaptive controller were implemented. Fuzzy logic was used to perform gain scheduling on the PD controller to improve its performance.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor Zheng Hong Zhu for his guidance and continuous support throughout my thesis research.

I would also like to thank my family for believing in me and supporting me through the difficult times.

Table of Contents

List of Tables

List of Figures

Symbols and Conventions

All units are given in SI units, time in seconds (s), distance in meters (m), frequency in Hertz (Hz), and velocity in meters per second (m/s).

All the vectors matrices are shown in bold italic with all the matrices. Matrices composed of other elements will have square brackets [ ].

List of Symbols

d_s	Waypoint separation distance
d_τ	Magnitude of displacement between capture point and end-effector
dx^{WF}	X distance between the chaser and target simulators
dy^{WF}	Y distance between the chaser and target simulators
F_e	External forces
I_{RW}	Moment of inertia of the reaction wheel
I_c	Moment of Inertia of the chaser simulator
$L_k \ k = 1,2,3$	Length of the k^{th} link in the manipulator
L_o	Robotic manipulator attachment offset
$L_{RW\max}$	Maximum angular momentum of the reaction wheel
$R_k \ k = 1,2,3$	Joint positions of the manipulator servo motor

T_o	Half the capture feature width
T_s	Half the base size of the target simulator
X_c^{CF}	X position of the target feature in the camera frame
X_R^{RF}	X end effector position in the robotic frame
X_{CP-EE}^{RF}	X distance between the capture point and the end-effector
$X_{Rk}^{RF}\ \ k=1,2,3$	X position for the R kth joint
X_{CS}^{WF}	X position of the chaser simulator in the world frame
X_{TS}^{BF}	X position of the target simulator in the chaser base frame
X_{TS}^{WF}	X position of the target simulator in the world frame
X^{WF}	X position waypoint for the chaser simulator
Y_{TS}^{BF}	Y position of the target simulator in the chaser base frame
Y_c^{CF}	Y position of the target feature in the camera frame
Y_R^{RF}	Y end effector position in the robotic frame
Y_{CP-EE}^{RF}	Y distance between the capture point and the end-effector
$Y_{Rk}^{RF}\ \ k=1,2,3$	Y position for the R kth joint
Y_{TS}^{WF}	Y position of the target simulator in the world frame
Y_{CS}^{WF}	Y position of the chaser simulator in the world frame
Y^{WF}	Y position waypoint for the chaser simulator
Zr^{WF}	Orientation waypoint for the chaser simulator

θ_c^{CF}	Orientation of the target feature in the camera frame
θ_{TS}^{BF}	Orientation of the target feature in the base frame
θ_R^{RF}	End effector orientation in the robotic frame
θ_{TS}^{WF}	Orientation of the target simulator in the world frame
θ_{CS}^{WF}	Orientation of the chaser simulator in the world frame
$\omega_{RW\max}$	Maximum angular velocity of the reaction wheel
$\omega_{c\max}$	Maximum angular velocity of the chaser simulator
$\omega_e \in R^3$	Angular velocity for the end-effector
$\omega_o \in R^3$	Angular velocity for the base spacecraft
φ^{WF}	Orbit offset between the chaser and target simulator

$(\mathbf{a}_i + \mathbf{b}_i) \in R^3 (i = 1...n)$	Length of the ith link
$\mathbf{b}_0 \in R^3$	Length of the base satellite from its center
$\mathbf{c}_b \in R^6$	Velocity dependent non-linear terms for the base spacecraft
$\mathbf{c}_m \in R^6$	Velocity dependent non-linear terms for the manipulator links
$\mathbf{C}$	X, Y and orientation mapping of thrusters
$\mathbf{e}$	Error vector for position and orientation

$\mathbf{F}$	Force vector for each pair of thrusters
$\mathbf{F}_b$	Forces and torque on the base spacecraft
$\mathbf{H}_b \in R^{6X6}$	Inertia matrix of the base spacecraft
$\mathbf{H}_{bm} \in R^{6\times n}$	Coupling inertia matrix for the base spacecraft and the manipulator
$\mathbf{H}_m \in R^{n\times n}$	Inertia matrix of the manipulator links
$\mathbf{J}_b$	Jacobian matrix dependent on the base
$\mathbf{J}_G$	Generalized Jacobian Matrix
$\mathbf{J}_m$	Jacobian matrix dependent on the manipulator
$\mathbf{k}_i \in R^3 (i = 1...n)$	Length of the base satellite from its center
$\mathbf{K}_p, \mathbf{K}_v$	Proportional and derivative gains
$\mathbf{M}$	Matrix of the mass and inertia of the simulator
$\mathbf{p}_n \in R^3 (i = 1...n)$	The position of J_n joint in the inertial frame
$\mathbf{q}$	Position and orientation vector
$\mathbf{r}_0 \in R^3$	The position of the base satellite in the inertial frame
$\mathbf{R}$	Thruster activation vector
$\mathbf{v}_e \in R^3$	Linear velocity for the end effector
$\mathbf{v}_o \in R^3$	Linear velocity for the base spacecraft

$\ddot{\mathbf{x}}_b = \begin{bmatrix} \mathbf{v}_o \\ \boldsymbol{\omega}_o \end{bmatrix} \in R^6$	Linear and angular acceleration for the base spacecraft
$\dot{\mathbf{x}}_e = \begin{bmatrix} \mathbf{v}_e \\ \boldsymbol{\omega}_e \end{bmatrix} \in R^6$	Linear and angular velocity for the end-effector
$\boldsymbol{\tau}$	Torques on the manipulator joints
$\ddot{\boldsymbol{\theta}} \in R^n$	Angular acceleration of the manipulator joints

List of Abbreviations

ACS	Attitude Control System
ADR	Active Debris Removal
AI	Artificial Intelligence
ARM	Acorn RISC Machines
ATV	Automated Transfer Vehicle
CAD	Computer Aided Design
CCD	Charge-Coupled Device
DARPA	Defense Advanced Research Project Agency
DC	Direct Current
DEOS	Deutsche Orbital Servicing Mission
Dextre	Special Purpose Dexterous Manipulator
DOF	Degrees of Freedom

ERA	European Robotic Arm
ESA	European Space Agency
ETS-VII	Experimental Test Satellite VII
EVA	Extravehicular Activity
FL	Fuzzy Logic
FREND	Front-end Robotics Enabling Near-term Demonstration
GEO	Geostationary Orbit
HEO	Highly Elliptical Orbit
HST	Hubble Space Telescope
HTTP	HyperText Transfer Protocol
HTV	H-II Transfer Vehicle
IADC	Inter-Agency Space Debris Coordination Committee
IR	Infrared
ISS	International Space Station
ITAR	International Traffic in Arms Regulations
JAXA	Japan Aerospace Exploration Agency
JC	Joint Controller
JEMIRMS	Japanese Experiment Module Remote Manipulator System
LEAP	Locomotion Enhancement via Arm Push-Off
LED	Light Emitting Diode
LEO	Low Earth Orbit
MDA	MacDonald, Detwiler and Associates

MF	Membership Function
NM	Notational Missions
NTUA	National Technical University of Athens
OBC	On-Board Computer
OOS	On-Orbit Servicing
PD	Proportional Derivative
PINOCCHIO	Platform Integrating Navigation and Orbital Control Capabilities Hosting Intelligence Onboard
PWM	Pulse Width Modulation
RAM	Random Access Memory
SFFR	Space Free-Flying Robot
SPIDER	Space Infrastructure Dexterous Robot
SPP	Russian Solar Power Platform
SSV	Satellite Servicing Vehicle
SUMO	Spacecraft for the Universal Modification of Orbits
UN	United Nations
XML	Extensible Markup Language

Chapter 1 INTRODUCTION AND JUSTIFICATION

Summary: In this chapter, we define the problem, justify the undertaking of the study and outline the method of approach adopted in achieving the set objectives. Furthermore, a summary of the layout of the thesis.

1.1 Background of On-Orbit Servicing Vehicles

The tremendous number of space debris in orbit and the risk of Kessler Syndrome have sparked the UN (United Nations) to create an initiative aimed at cleaning space debris [1,2]. The initiative states that future space missions must comply with IACC (Inter-Agency Space Debris Coordination Committee) guidelines to reduce their environmental impact over the entire life cycle of the mission. In addition, new technologies to reduce the current debris population must be developed [3]. There has been great consideration on the types of capture methods that would be used for such a mission, for instance, robotic arms, nets or harpoons. Two missions have been formulated by the ESA (European Space Agency) to test technologies for active debris removal; RemoveDEBRIS and e.Deorbit [4,5].

RemoveDEBRIS was a demonstration mission that used a net and harpoon to capture debris of uncooperative targets in 2018 [4]. However, a great challenge exists due to the high launch and operational cost of the capture mission. These missions could cost up to €10 million for a single capture [4]. This poses the challenge of who should be paying for the development and de-orbiting of space debris.

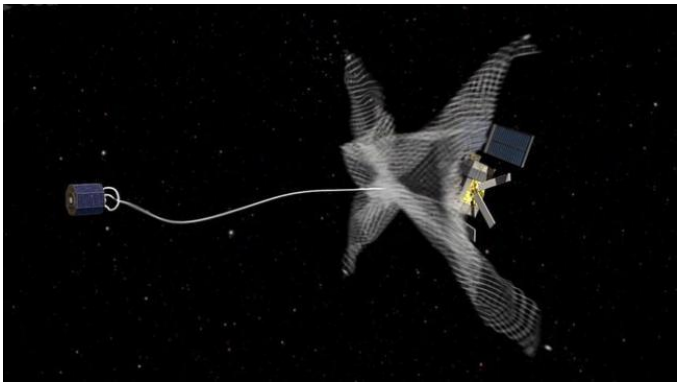


Figure 1.1 Illustration of RemoveDEBRIS using a net to capture a target [6].

Following many studies the ESA has made the decision to widen the scope of the Clean Space initiative to not only perform active debris removal but for the satellites to also become a space servicing vehicle [5]. This would produce a vehicle that can perform multiple tasks such as refueling, upgrading, repair and maneuvering of existing space objects, as well as the construction of large space objects. The future e.Deorbit mission will test and demonstrate robotic serving technologies. The consideration of both satellite servicing and active debris removal is important given the growth of private industry into space.

The European Space Agency is not the only entity looking towards OOS as a solution for refueling, upgrading, repair and maneuvering of existing space objects. Many private entities are looking towards this technology for the de-orbiting and repositioning of their satellites. With increased number of spacecraft launched, the number of satellites that would experience on-orbit failures will also increase. There is a 3-4% chance that a spacecraft will experience a total or partial failure in the first 30 mission days [7]. The cumulative cost of these failures account for a total of $4.4 billion dollars in losses between the years of 1990 and 2006 [8]. A failed satellite can threaten an entire constellation and a servicing vehicle can be used to repair or remove the satellite from the constellation. The servicing vehicle can be contracted to de-orbit, repair, and refuel satellites. This allows satellite owners to save on the cost of sending a replacement satellite.

On-orbit servicing (OOS) technologies has been a topic of investigation since the early 80's [9] for its ability to repair, re-fuel, assemble and de-orbit space objects. A typical satellite servicing vehicle (SSV) consists of a base spacecraft with attached n-DOF (Degrees of Freedom) manipulators [10], see Figure 1.2. The base satellite is used to maneuver in space to perform orbital transfers and rendezvous with target space objects. The robotic manipulators are used to capture and manipulate the target space object.

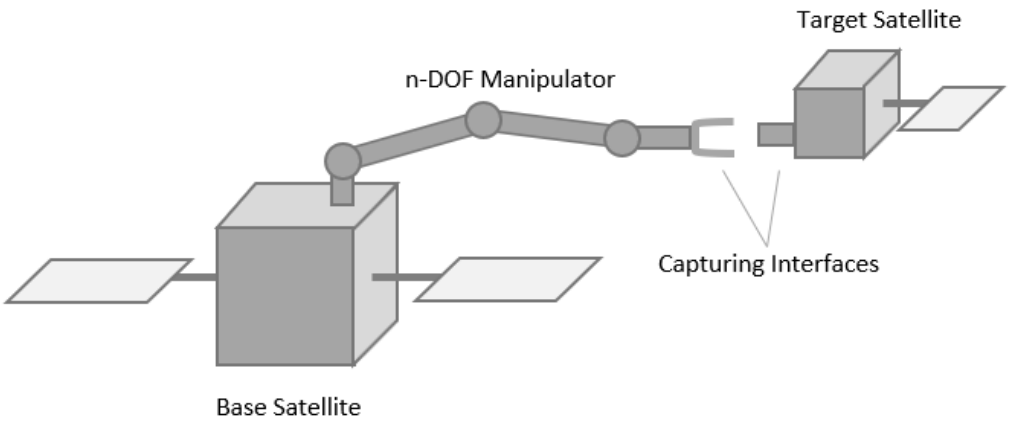


Figure 1.2 Illustration of base spacecraft with a robotic manipulator.

1.1.1 Mission History

In this section, spacecraft with robotic manipulators and their missions that were planned and/or launched are reviewed. The robotic manipulators on landers/rovers and robots used inside the ISS are excluded from review as they are outside the scope of this paper.

Developed by the Canadian Space Agency (CSA) and MDA, the Shuttle Remote Manipulator System (SRMS) also known as the Canadarm was used on the space shuttle program to deploy and capture satellites, position astronauts and move cargoes [11] from 1981-2011 [12,13] when the space shuttle program was retired. The manipulator was 15 metres in length and had 6-DOF [10,14]. Following the Canadarm, the Canadarm2 was installed on the ISS in 2001 and is currently in operation. The Canadarm2 assisted the construction of the ISS, performs maintenance, moves equipment and astronauts and

handles payloads. What makes the Canadarm2 unique is its ability to attach to different ports on the ISS' exterior [11] and move itself to the different ports. The Canadarm2 is a 7-DOF manipulator with 17 meters reach [14]. The Candarm3 is currently under development and planned for the future Lunar Gateway project. The Lunar Gateway will begin construction in 2022 with its completion in 2026 [15]. The proposed design of the Canadarm3 is 8.5m in length with 7-DOF [14].

In 2008, the Special Purpose Dexterous Manipulator (Dextre) was installed on the ISS. Dextre consists of two robotic manipulators each 3m in length and 7-DOF [11]. Dextre can be moved around at the end of the Canadarm and possesses the ability to install and replace equipment on the space station. The end-effectors posses a retractable motorized wrench, a camera with lights and a retractable connector to provide power, data and video [16].

Another robot system installed on the ISS is the Japanese Experiment Module Remote Manipulator System (JEMIRMS) built by the Japan Aerospace Exploration Agency (JAXA). This robot has a 6-DOF 10m long main arm and a 6-DOF 2m arm for more dexterous tasks [10,17,18].

Another arm that will be attached to the ISS is the European Robotic Arm (ERA) developed by the European Space Agency (ESA). The robotic arm is 11 meters long with 7-DOF. It has two booms and a relocatable base [10,17]. The manipulator was responsible for the installation of solar panels on the Russian Solar Power Platform (SPP), moving payloads and other maintenance tasks. The ERA can be operated either inside or outside the ISS by crewmembers [17].

The first remotely controlled robot was ROTEX during the Spacelab D2 mission in 1993. The manipulator flew on the Columbia space shuttle. ROTEX has a 1m length and demonstrated multisensory gripper technology, feedback control concepts and 3D simulation in the telerobotic ground station [19].

In 1997, the ETS-VII mission was launched to demonstrate the autonomous rendezvous and docking, and other robotic servicing functions. It was the first experimental test vehicle to demonstrate the capture and repair of a target satellite. The servicing spacecraft consisted of a 2-meter long robotic arm with 6-DOF. The mission demonstrated teleoperation with a ground station, robotic servicing tasks and autonomous capture of a target satellite [9,10].

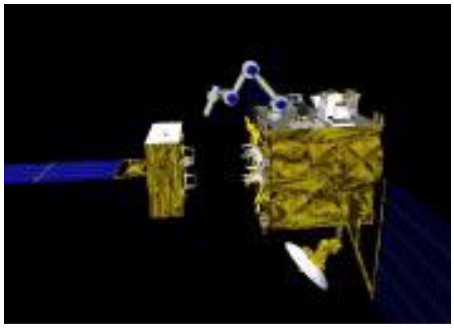


Figure 1.3 ETS-VII simulation created by Tohoku University [9].

The Orbital Express mission successfully demonstrated on-orbit servicing of a target spacecraft in 2007. The mission was developed by the Defense Advanced Research Project Agency (DARPA) and Boeing. The servicing robot successfully performed rendezvous and docking autonomously. The manipulator then transferred a supplementary battery and backup computer [10].

The Spacecraft for the Universal Modification of Orbits (SUMO) was a testbed to perform several on-orbit servicing capabilities which was later renamed to Front-end Robotics Enabling Near-term Demonstration (FREND). The manipulator was developed in 2005 which had 7-DOF [20]. The length of FREND was 2m. The manipulator had a mass of 157kg including its electronics [21]. The testbed successfully demonstrated autonomous docking and capture, and serviced the target [18]. The FREND manipulator is listed here as it was the proposed manipulator for the Notational Missions (NM) [21] and for the PHOENIX OOS program.

The NM are a set of concept missions developed by NASA in 2010 to study satellite servicing. The missions were GEO Supersync (NM1), GEO Refueling (NM2), LEO Refurbishing (NM3), Earth-Moon L1 (EML1), Robotic Assembly (NM4), High Elliptical Orbit Human/Robotic Refurbishing (NM4) and Sun-Earth L2 Human/Robotic Assembly (NM6) [18]. The missions make use of the FREND robot arm and the Ranger system [21].

The DARPA PHOENIX program started in 2012 and was planned for launch in 2015. Using the FREND manipulator, a cooperative satellite would use its components to make a new free-flying space system [10,18,22]. Derived from the Phoenix project, DARPA began the Dragonfly mission in 2015 with an updated objective of Phoenix focusing on on-orbit robotic assembly of a GEO communications satellite [18,23]. Once out of the ground demonstration phase, the manipulators were again re-named to Space Infrastructure Dexterous Robot (SPIDER) to be attached to the Restore-L vehicle. Restore-L is a servicing vehicle designed to service LEO satellites, with a specific target Landsat 7 [24]. Restore-L is designed to perform refueling, relocation and redeployment of LEO

spacecraft using robotic servicing technology [18]. Restore-L is planned to launch in 2020 [24] but was extended to 2023 [25]. Including SPIDER Restore-L will consist of 3 robotic manipulators. Two manipulators to perform OOS functions [26] and the SPIDER manipulator (5m) to perform on-orbit assembly of a 3m Ka-band communication antenna [25].

The Deutsche Orbital Servicing Mission (DEOS) was another OSS demonstration mission aimed to evaluate techniques for rendezvous, capture and de-orbit noncooperative tumbling satellites using a robotic manipulator [10]. A typical non-cooperative target would tumble at rotations 4 deg/s and greater [27,28]. This mission had a planned launch date of 2018, however, was canceled [29].

e.Deorbit is an active debris removal mission that is planned for 2025 [30]. The concept under consideration is to capture a debris using a net and a robotic arm. The goal is to autonomously capture and deorbit an uncooperative target. This will be the first mission to attempt an automated capture of an uncooperative target. The mission also motivates further research in capture mechanism, guidance and navigation, image recognition and onboard processing [31]. The manipulator on e.Deorbit will be 4.2 m in length [32]. The mass of the base spacecraft is expected to be 1,600 kg [33].

A recent development is the private company Altius Space Machines who is developing several robotic capturing mechanisms and OOS vehicles, funded by NASA and the Department of Defence [34]. In December of 2019, OneWeb, a company currently launching a large scale Low Earth Orbit (LEO) constellation, teamed up with Altius Space

Machines to install their grappling fixture on their constellation satellites to perform deorbiting as part of their responsible space initiative [35].

In the following table, all the above missions with their base spacecraft and manipulator parameters are listed.

Table 1.1 On-orbit robotic mission data

Mission	Launched	DOF	Size of Base Spacecraft (m)	Arm Length (m)	Spacecraft Mass (kg)	Arm Mass (kg)
Canadarm	1981-2010 (Space Shuttle) [11–13]	6 [2,24,26,34]	37.25x23.8x17.27 [36]	15 [10,11,14]	78,000 [36]	410 [11,14]
Canadarm2	2001 (ISS) [11–14]	7 [10,12,14,22]	109x73 [37]	17 [10,14]	408,000 [37]	1497 [14]
Rotex	1993 (Space Shuttle) [12]	6 [10,12]	37.25x23.8x17.27 [36]	1 [19]	78,000 [36]	-
ETS-VII	1997 [12]	6 [12]	2.6x2.3x2 [38]	2 [38]	2,450 [38]	45 [38]
Orbital Express	2007 [12,18,39]	6 [12,18,22,40]	1.78m x 1.65m x 1.8m [18]	3 [18,40]	953 kg [18]	71 [18,40]
Dextre	2008 (ISS) [11]	3 [41]	109x73 [37]	3.7 [16]	408,000 [37]	1710 [16]

Japanese Experiment Module Remote Manipulator System	2009 (ISS) [12]	6 [10]	109x73 [37]	10 and 2 [10]	408,000 [37]	-
GEO Supersync (NM1)	Was planned for 2015 [18]	7 (FREND) [18,20]	-	2 [20,21]	3,694 [18]	77 [20]
GEO Refueling (NM2)	Was planned for 2015 [18]	7 (FREND) [18,20]	-	2[20,21]	1,894 [18]	77 [20]
Phoenix Program	Planned 2015 (Now Dragonfly) [18,42]	7 [18,20]	-	1.8 [43]	-	77 [20,43]
LEO Refurbishing (NM3)	Was planned for 2017 [18]	3 [18]	-	14 and 7 [18]	-	-
DEOS	Canceled (2018) [29]	7 [44]	2.6x1.7x1.8 [18,45]	3.232 [44]	786 [18]	40.5 [46]
Dragonfly	Planned 2020 (Now SPIDER on Restore-L) [18]	-	-	-	-	-
European Robotic Arm	Under Review [47]	7 [10,17,47]	109x73 [37]	11.3 [10,17,47]	408,000 [37]	630 [47,48]

Canadarm3	Planned 2022 (Lunar Gateway) [14,15]	7 [14]	-	8.5 [14]	-	715 [14]
Restore-L	Planned 2023 [25]	-	-	5 [25]	-	-
e.Deorbit	Planned 2025 [30]	7 [32]	-	4.2 [32]	1,600 [33]	-
EML1 Robotic Assembly (NM4)	Planned for 2025 [18]	-	-	20 and 6 [18]	6,592 [18]	-
HEO Human/Robotic Refurbishing (NM5)	Planned 2035 [18]	-	-	20 and 2 [18]	31,870 [18]	-
Human/Robotic Assembly (NM6)	Planned 2035 [18]	-	-	-	27,000 [18]	-

1.2 Justification of the Research

Satellites with robotic manipulators are the encompassing solution to both active debris removal and servicing tasks. Space based manipulators have been used throughout history to perform capture and repair of satellite; however, a specific class of space-based manipulators have become the future of active debris removal and satellite servicing tasks, requiring extensive research to bring to fruition.

Traditionally, manipulators such as the Canadarm and Canadarm2, on the Space Shuttle and the International Space Station (ISS), have been used to perform capture and repair of spacecraft. Due to the mass of the base spacecraft being several orders of magnitude larger than that of the robotic manipulator, the disturbances from the manipulator onto the base are considered negligible. However, servicing spacecraft must have low mass to reduce development and launch costs. This creates a class of spacecraft where the mass of the manipulator is comparable to that of the base satellite. This imposes the challenge of having significant disturbance onto the base satellite due to the motion of the attached robotic manipulator. This creates a complex dynamic coupling that must be considered when developing control and path-planning algorithms for satellite servicing vehicles.

OOS vehicles are subject to unique motion and dynamics that make the control difficult in comparison to their earthbound fixed based counterparts. The base of the spacecraft moves due to the action-reaction principle of momentum conservation. Path planning and control algorithms must be designed to compensate for the moments induced by the manipulator motion, while still being efficient or executable for the limited computational performance of space grade computer hardware.

In addition to the complex dynamics of OOS, there are many technical features required for the success of each phase of an OOS missions that are still actively investigated and developed. In order to test and verify these technical features on Earth before launch into space, testbeds must be developed to simulate the microgravity environment as well as having hardware that is comparable to hardware being used on the OOS vehicle.

1.2.1 The Dynamics of Satellite Servicing Vehicles

Satellites with robotic manipulators are subject to unique motion dynamics that make control difficult in comparison to their Earthbound fixed base counterparts. Free-Floating/Flying manipulators are subject to both kinematic and dynamic singularities whereas fixed base robotic manipulators are only subject to kinematic singularities.

The equation of motion for a free-flying space robot is derived by the Lagrangian method [9,49,50].

$$\begin{bmatrix} \mathbf{H}_b & \mathbf{H}_{bm} \\ \mathbf{H}_{bm}^T & \mathbf{H}_m \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{x}}_b \\ \ddot{\boldsymbol{\theta}} \end{bmatrix} + \begin{bmatrix} \mathbf{c}_b \\ \mathbf{c}_m \end{bmatrix} = \begin{bmatrix} \mathbf{F}_b \\ \boldsymbol{\tau} \end{bmatrix} + \begin{bmatrix} \mathbf{J}_b^T \\ \mathbf{J}_m^T \end{bmatrix} F_e \tag{1.1}$$

where

$\mathbf{H}_b \in R^{6X6}$: Inertia matrix of the base spacecraft

$\mathbf{H}_m \in R^{n\times n}$: Inertia matrix of the manipulator links

$\mathbf{H}_{bm} \in R^{6\times n}$: Coupling inertia matrix for the base spacecraft and the manipulator

$\mathbf{c_b} \in R^6$: Velocity dependent non-linear terms for the base spacecraft

$\mathbf{c_m} \in R^n$: Velocity dependent non-linear terms for the manipulator links

$\ddot{\mathbf{x}}_b = \begin{bmatrix} \mathbf{v}_o \\ \boldsymbol{\omega}_o \end{bmatrix} \in R^6$: Linear and angular acceleration for the base spacecraft

$\ddot{\boldsymbol{\theta}} \in R^n$: Angular acceleration of the manipulator joints

$\mathbf{F}_b$: Forces and torque on the base spacecraft

F_e : External forces

$\boldsymbol{\tau}$: Torques on the manipulator joints

$\mathbf{J}_b$: Jacobian matrix dependent on the base

$\mathbf{J}_m$: Jacobian matrix dependent on the manipulator

The general model for a space robot system with a single manipulator is shown in Figure 1.4. The manipulator consists of the base B_0 and B_i links. J_i is the joint connecting links B_{i-1} and B_i. The system is described as an n+1 serial chain with n active joints.

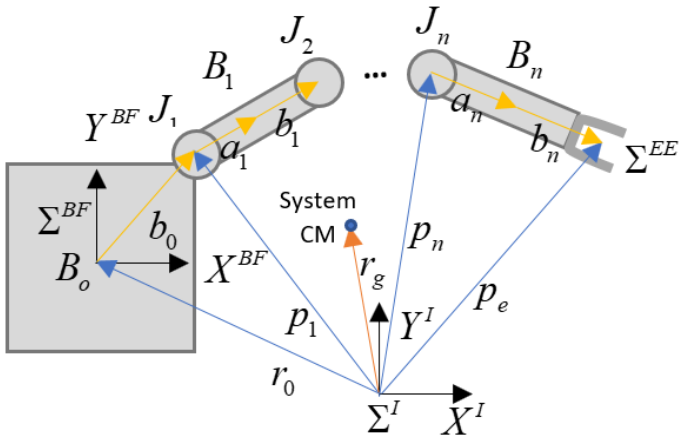


Figure 1.4 General model for a space robot with a single manipulator [49,50].

Here

$\mathbf{p}_n \in R^3 (i = 1...n)$: The position of J_n joint in the inertial frame

$\mathbf{r}_0 \in R^3$: The position of the base satellite in the inertial frame

$(\mathbf{a}_i + \mathbf{b}_i) \in R^3 (i = 1...n)$: Length of the ith link

$\mathbf{b}_0 \in R^3$: Length of the base satellite from its center

$\mathbf{k}_i \in R^3 (i = 1...n)$: Length of the base satellite from its center

$\ddot{\mathbf{x}}_e = \begin{bmatrix} \mathbf{v}_e \\ \boldsymbol{\omega}_e \end{bmatrix} \in R^6$: Linear and angular acceleration for the end-effector

Summing all the links from the base spacecraft's center of mass in the inertial frame ***I,*** we obtain the position of the end effector as.

$$\mathbf{p}_e = \mathbf{r}_0 + \mathbf{b}_0 + \sum_{k=1}^{n} (\mathbf{a}_k + \mathbf{b}_k) \tag{1.2}$$

Differentiating Eq. (1.2) yields a relationship between the end-effector's linear and angular joint velocities. We also obtain a relationship between the end-effector's angular velocity and joint velocities.

$$\mathbf{v}_e = \mathbf{v}_o + \boldsymbol{\omega}_o \times (\mathbf{p}_e - \mathbf{r}_o) + \sum_{k=1}^{n} [\mathbf{k}_k \times (\mathbf{p}_e - \mathbf{p}_k)] \dot{\boldsymbol{\theta}}_k \tag{1.3}$$

$$\boldsymbol{\omega}_e = \boldsymbol{\omega}_o + \sum_{k=1}^{n} \mathbf{k}_k \dot{\boldsymbol{\theta}}_k \tag{1.4}$$

Re-arranging in matrix form we obtain the differential kinematic equation of the combined base manipulator system,

$$\ddot{\mathbf{x}}_e = \mathbf{J}_b \ddot{\mathbf{x}}_b + \mathbf{J}_m \dot{\boldsymbol{\theta}} \tag{1.5}$$

In the free-floating scenario, there are no external forces or torques on the system. Linear and angular moments are conserved.

In the free-floating scenario, the external forces/torques on the base are assumed to be zero ($F_e = 0$). The motion of the satellite with the robotic manipulator is dependent only on the internal forces and torques of the manipulator joints $\boldsymbol{\tau}$. Integrating the first line of Eq. (1.1), we obtain the equation that describes the total momentum of the free-floating system around its center of mass [9].

$$\mathbf{H}_b \dot{\mathbf{x}}_b + \mathbf{H}_{bm} \dot{\boldsymbol{\theta}} = 0 \tag{1.6}$$

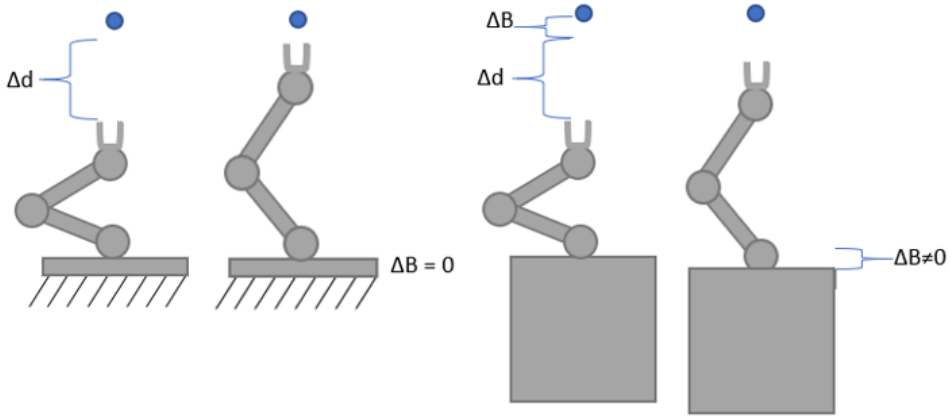


Figure 1.5 Illustration of the transfer of linear momentum to the base spacecraft.

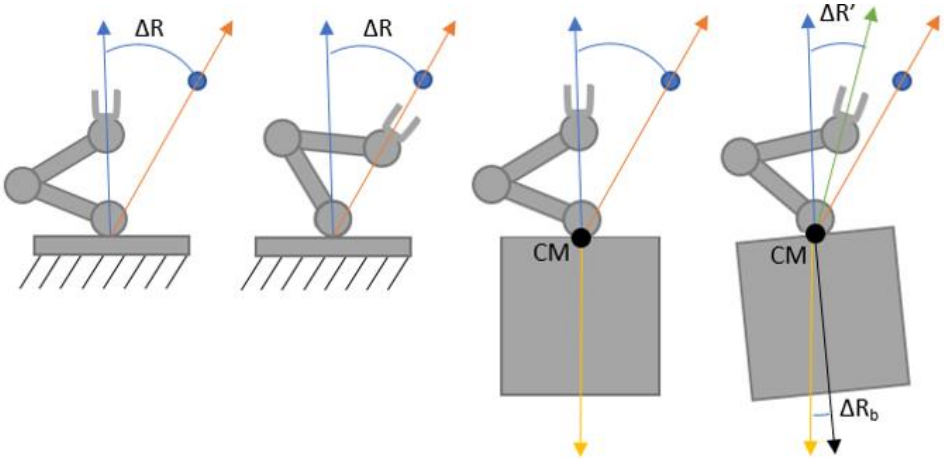


Figure 1.6 Illustration of the transfer of angular momentum about the CM of free-floating system.

Substituting the differential kinematic equation (1.5) for the end effector linear and angular velocity with the momentum conservation equation (1.6), we obtain a direct relationship between the end-effector linear and angular velocity and its joint rates.

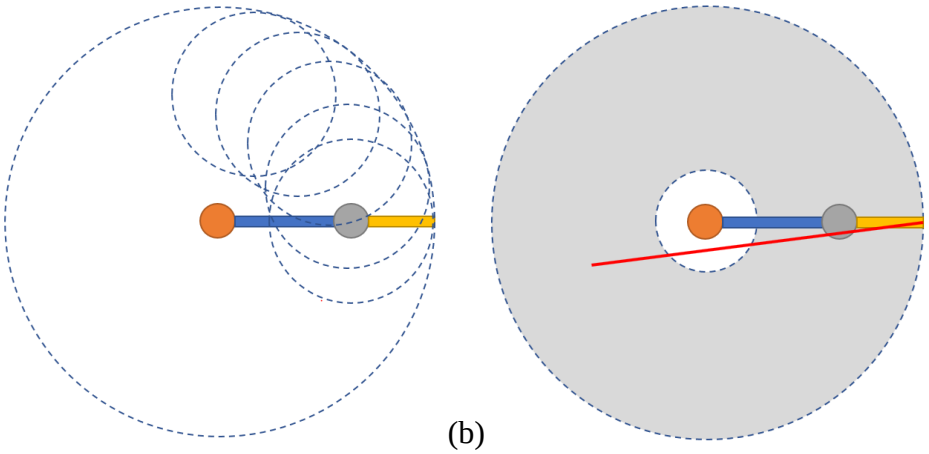
$$\begin{bmatrix} \mathbf{v}_e \\ \boldsymbol{\omega}_e \end{bmatrix} = \left[\mathbf{J}_m + \mathbf{J}_b \mathbf{J}_{bm} \right] \dot{\boldsymbol{\theta}} = \mathbf{J}_G \dot{\boldsymbol{\theta}} \tag{1.7}$$

$$\mathbf{H}_b^{-1} \mathbf{H}_{bm} = \mathbf{J}_{bm} \tag{1.8}$$

This transformation $\mathbf{J_G}$ is called the Generalized Jacobian Matrix. This method requires that we know the linear and angular velocities of the end-effector, then using the inverse Generalized Jacobian Matrix to solve the joint rates. However, with this method the system is subject to dynamic singularities in the Generalized Jacobian Matrix [49] as well as the kinematic singularities commonly encountered in robotic manipulators.

Kinematic singularities are a problem due to the inverse mapping from cartesian space to joint space. There are two kinds of kinematic singularities. The first is boundary/workspace singularities where if you tell the system to reach beyond where the manipulator can be positioned. The manipulator would try to reach outside of its operable work space [51,52].

For example, in Figure 1.7, we have a 2-DOF planar robotic manipulator. The red line indicates the desired path of the endpoint. The grey area, shown in Figure 1.7(b), indicates the manipulator's reachable workspace. The desired path crosses the region in the middle where the endpoint cannot be reached, and the inverse kinematics cannot be solved. At this point a workspace kinematic singularity is encountered.



(a) (b)

Figure 1.7 Illustration of robotic manipulator workspace [51].

The second is the internal kinematic singularity. An example of internal kinematic singularities is depicted in Figure 1.8. In Figure1.8(a), we have a 2-DOF planar robotic manipulator that is commanded to follow the dotted red path. The limiting factor here is that the second joint cannot have an angle less than zero degrees. As the endpoint follows

the path it reaches the point where the second joint reaches zero degrees. To continue its path, the first joint must rapidly rotate 180 degrees depicted in Figure 1.8(c). At the same time, the second joint must rapidly change from decreasing its angle to increasing as shown in Figure 1.8(d). This causes the infinite joint velocities, which can damage the manipulator and cause unpredictable motion.

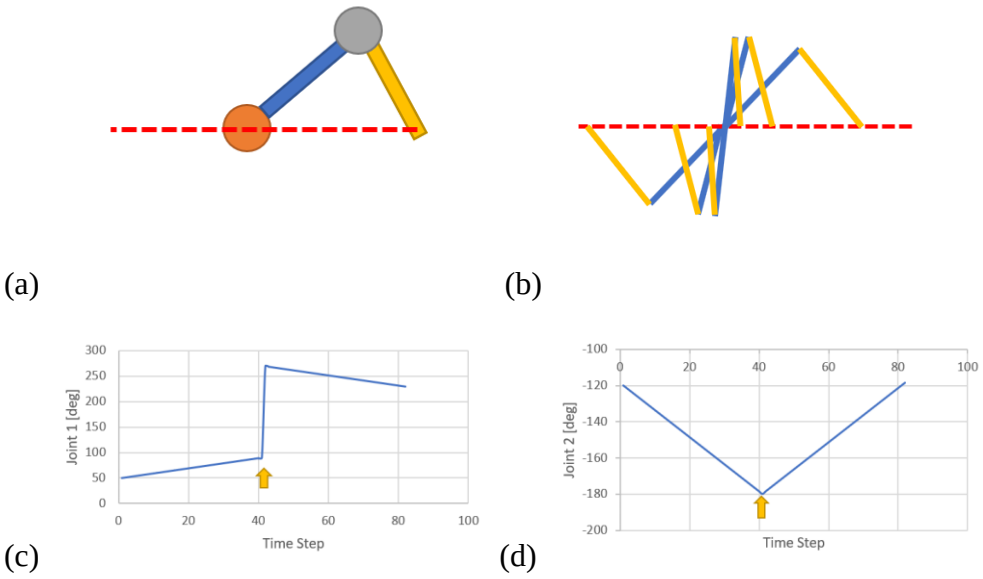


Figure 1.8 Depiction of internal kinematic singularity [51].

Both boundary and internal singularities can be predicted beforehand by analyzing the workspace and arm properties. With this known information, theses singularities can be avoided.

Dynamic singularities on the other hand are a function of the dynamic parameters of each body (for example the inertial parameters of the links) [53,54]. They are dependent

on the path. These singularities are a result of the Generalized Jacobian Matrix losing rank [50].

These singularities cannot be predicted beforehand when using the Generalized Jacobian Matrix. In this method, the way the joints move is not pre-planned and future joint positions are not known. Testbeds that can accurately mimic the microgravity environment so that the dynamics that an OOS vehicle would face can be re-created here on Earth.

1.2.2 Servicing Satellite Tasks

There are several tasks [9] in a servicing satellite mission. Each task's current development and future research goals are also summarized.

1. **Initial Approach**: Far-range approach, using orbital mechanics. This step has been successfully executed in many space missions. Current research is in obtaining high precision orbital positioning without complex sensors. Currently, sensor fusion using many low-cost sensors is one method being actively investigated.

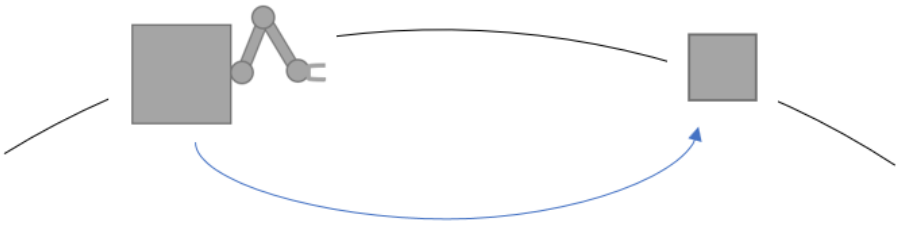


Figure 1.9 Far-range approach, using orbital mechanics.

2. **Fly-around and Inspection**: A fly-around is a useful maneuver to observe a target to assess damage, mounting points and target motion. Current research methods are focused in the determination of the target motion using a CCD camera and a designated target on the target space object. Investigation of determining the target motion, without a designated target, on the target space object is actively being investigated.

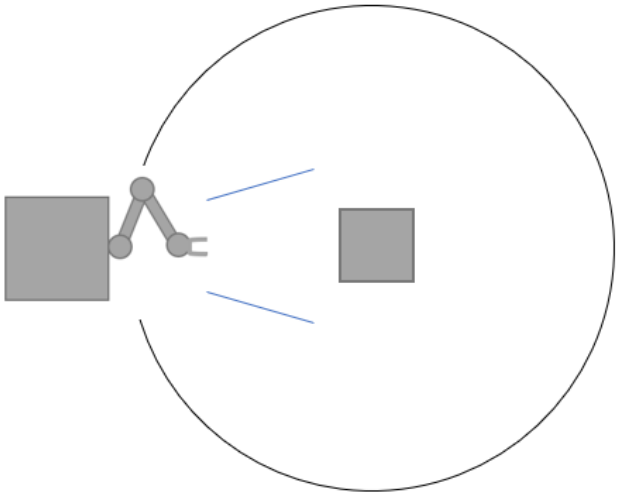


Figure 1.10 Fly-around and visual inspection.

3. **Final Approach and Rendezvous**: This stage has been successfully demonstrated by Experimental Test Satellite (ETS) VII, Orbital Express and many non-manipulator spacecraft such as the Automated Transfer Vehicle (ATV) [55]. Experimental Test Satellite (ETS) VII and Orbital Express missions had well defined targets and sensors on the target spacecraft. The European Space Agency [56] and many private companies [57] are actively investigating standardized

passive markers. More research is needed for targeting of non-specified, general features to measure the relative motion and distance of a target spacecraft.

Figure 1.11 Final approach and rendezvous.

4. **Course Manipulation, Visual-Servo Tracking, and Capture:** The goal of this phase is to use a vision or sensor system to guide the manipulator to capture the target. This has been successfully demonstrated by ETS-VII and the Orbital Express. An important consideration during the capture phase is the attitude control system should be turned off [9]. This is not a requirement, but it is highly recommended. The reason for turning off the attitude control system (ACS) is the contact forces during the capture can be considered random. Turning off the ACS prevents random triggering of the ACS which would lead to shock and damage of the spacecraft and the target [27,58]. This operation mode also leads to fuel savings and therefore extends the overall duration of the mission.

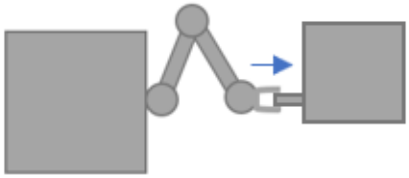


Figure 1.12 Visual servo tracking, course manipulation and capture.

To capture a target using a manipulator, precise positioning is required. Motion of the arm causes reactions to disturb the attitude of base spacecraft. During the rendezvous, this is highly undesirable when such precise motion and control is required. A large area of research is the compensation for such undesirable motions in the control of arm manipulation.

In addition to how to capture the target, a large area of research is in creating capture mechanisms and target capture features. The Clean Space Initiative in 2012 proposed the development of a passive interface on a target satellite and the investigation of a corresponding capture mechanism [1]. These interfaces are only applicable to new space vehicles being launched. A robust capture mechanism is required to capture existing space objects that do not have well defined capture interfaces. This problem has proven to be challenging and is currently under investigation. For capturing without a robotic manipulator for the purpose of refueling, ViviSat claimed that their interface could dock with 90% of geostationary satellites in orbit [59] while a competitor MDA can dock with 75% of geostationary satellites in orbit [60].

5. **Impact Dampening and Motion Suppression:** Minimizing impact forces is important to reduce damage on the robotic manipulator. Both impact dampening and motion suppression is especially important for capturing a tumbling target.

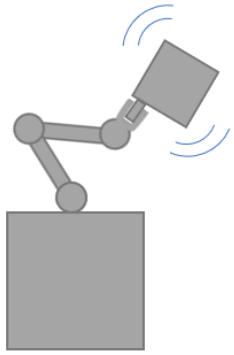


Figure 1.13 Impact dampening and motion suppression.

Optimal methods of detumbling a docked space object using onboard thrusters have been investigated since the Space Shuttle [61]. Optimal methods to detumble a target using only the manipulator(s) for fuel conservation is an area of active research. The manipulator(s) would rotate in such a way to counteract the momentum of the system caused by the tumbling target space object [27].

6. **Target Berthing:** Target berthing has been successfully demonstrated on many missions such as SpaceX Dragon capsule docking to the ISS [62] and the Mission Extension Vehicle (MEV-1) to IntelSat-901 [63]. There is the increased challenge of performing breathing using the robotic arm to dock the spacecraft. The vibration of the manipulator becomes an issue due to the large payload at the endpoint and

the flexibility of the manipulator arm. This has been demonstrated on the Space Shuttle when capturing the Hubble Space Telescope (HST) [64], ISS when capturing cargo such as HTV [65,66], ETS-VII and Orbital Express missions.

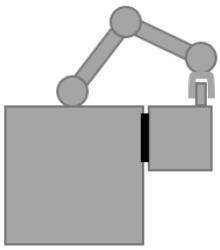


Figure 1.14 Target berthing.

7. **Orbital Exchange and Refueling:** Exchange of components and refuelling of the target spacecraft have been verified by the Space Shuttle, ISS and experimental OOS missions ETS-VII and Orbital Express.

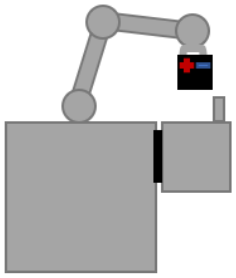


Figure 1.15 Orbital unit exchange and refueling.

8. **More Dexterous/Skilled Manipulation:** Through teleoperation, fine handling of wires and connectors have been tested in the ETS-VII mission and demonstrated

using the Dextre robotic manipulator onboard the ISS on mock satellites [67]. Further research is needed as well as accomplishing this task autonomously rather than through teleoperation and pre-programmed scripts.

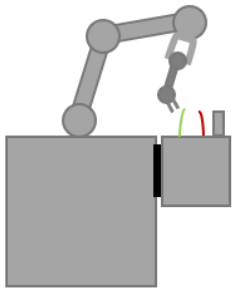


Figure 1.16 More dexterous manipulation.

1.3 Limitations of Existing Approaches

Existing earthbound methods used to test OOS technology are incapable of accurately simulating the microgravity environment. Hardware-in-the-loop systems use industrial robotic manipulators and simulated software to test OOS technology. Developed OOS simulation software, used in hardware-in-the-loop testbeds are subject to the accuracy of mathematical models and the accuracy of the industrial robots [68,69].

Other testbeds are weight-reducing systems which use wires to suspend the manipulator; parabolic fights where a plane flies downwards placing the contents in a free-fall mimicking the microgravity environment for a short duration; and drop towers that allow objects to free-fall mimicking the microgravity environment.

The hardware-in-the-loop, weight-reducing suspension systems, parabolic flights and drop towers have limited range of operation and the size of the testable workspace

[69]. Parabolic flights and drop towers have limited operational times for their testbeds. With parabolic flights it is possible to achieve 25 seconds of free fall while for drop towers most facilities free fall lasts for 10 seconds [69,70].

Neutral buoyancy testbeds submerge hardware in a pool allowing them to free float. Neutral Buoyancy simulators suffer from adding unwanted resistance to moving parts and requiring hardware to be waterproof as they are submerged in a pool to replicate the microgravity environment [69].

Air-bearing tables are also limited in their range of motion. Air-bearing tables are limited to two translational and one rotational component whereas as spacecraft has a total of 6-DOF. Air-bearing simulators have the highest equivalence to the microgravity environment [71].

1.4 Objectives of the Research

Currently, there are no self-contained testbeds on which OOS can be tested. All other testbeds rely on external sensors and computers to provide the positioning and attitude information of the spacecraft and the target satellite. The next chapter will review existing testbeds, their strengths and limitations, and why they are not self-contained. The objectives of this work are to:

(i) develop a self-contained testbed for OOS development;

   a. develop a vision system for the chaser simulator to track the target simulator,

b. develop a PD controller for the chaser and target simulators,

c. develop a 3-DOF planer manipulator for the chaser simulator,

d. develop a capture mechanism and target capture fixture,

e. develop a path planning algorithm for final approach and partial fly-around maneuvering,

f. develop a detumbling algorithm for post-capture stabilization.

(ii) validate the versatility of the testbed;

a. perform capture on a stationary free-floating target,

b. perform capture and detumbling on a free-rotating target,

c. develop and implement an improved adaptive controller using fuzzy gain scheduling for the above tasks.

1.5 Method of Approach

The outline of the methodology to achieve the above stated objectives is shown in Figure 1.17.

Air-bearing tables are also limited in their range of motion but have the highest equivalence to the microgravity environment. What makes this air-bearing testing system different from existing systems is that the target and chaser simulators are completely self-contained systems like the spacecraft in orbit. Thus, the simulators' position and orientation are sensed and computed using on-board hardware. The target's position and orientation are computed using a monocular camera on the chaser simulator and is computed onboard.

This is different from existing systems that use external sensors and computers to compute the position and orientation of both simulators and then transmit the relevant data to the simulators.

A robotic manipulator was developed with an appropriate gripper on the chaser and corresponding docking probe on the target. Path planning algorithms were implemented to dynamically perform the shortest straight-line path to the target and then perform a partial fly-around maneuver to align with the target docking face. A PD controller was designed and its stability with optimal gains was proved. The controller was then implemented on both the chaser and target simulators. The total working system was then validated by performing the capture of a stationary target and the capture and detumbling of a rotating target. Finally, to demonstrate how the system can be used to test different algorithms, an Adaptive Controller using Fuzzy-Logic to perform gain scheduling on a PD controller was proposed, implemented and tested.

Figure 1.17 Outline of approach methodology.

1.6 Layout of Thesis

This document contains seven chapters. Following the introductory Chapter 1, Chapter 2 provides a detailed review of on-orbit technology in three main areas: (1) all launched and planned on-orbit spacecraft that user robotic manipulators, (2) different

ground based experimentation and verification methods, and (3) a review of other air-bearing testbeds with robotic manipulators. Chapter 3 gives a detailed description of the hardware and software developed for the satellite simulator pair. Chapter 4 outlines the two control strategies implemented on the simulator to demonstrate its modularity in its ability to implement different hardware and software solutions. In Chapter 5, the simulators and associated algorithms were tested using a Matlab & Simulink simulation to further tune the controllers. Chapter 6 shows the results of the PD controller system and the improvement observed from the adaptive controller. Finally, in Chapter 8, we conclude the work, identify the original contributions of the thesis, and outline the directions for future work and investigation.

Chapter 2 LITERATURE REVIEW

Summary: The literature review covers two main sections. the first section outlines different ground-based methods to simulate the microgravity environment for experimentation and verification; and the second outlines other air-bearing simulators with robotic manipulators.

2.1 Ground Based Experimentation and Verification

Many approaches have been developed for the ground validation of space robotics. The key aspect for these ground experiments is to simulate the microgravity environment. The first is to test underwater and make all equipment neutral buoyancy, see Figure 2.1. The drawback is the high resistance (drag) from the test medium. The equipment also requires waterproof hardware to operate [69]. This facility is primarily used for astronaut training to perform extravehicular activates (EVA).

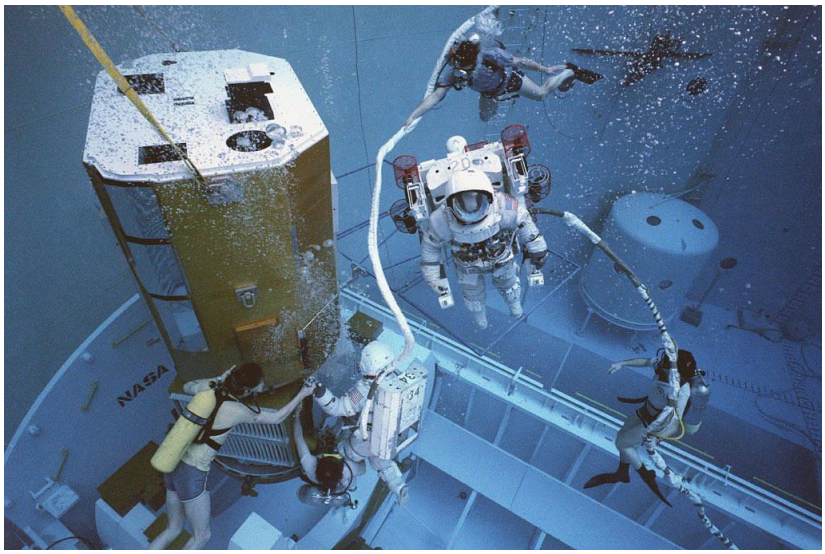


Figure 2.1 NASA Neutral Buoyancy Simulator - Solar Max Testing [72].

A second method is to use a weight-reducing system to lift the weight caused by gravity on the robotic hardware. This system limits the motion of the test hardware. Another popular testbed is the use of industrial robots where the motions of the chaser and target satellite are executed on the robotic arms, see Figure 2.2. This implementation allows for the simulation of the full 6-DOF and allows for the force contact feedback on the end-effector. The satellites are simulated in software and the resultant motion is executed by the industrial manipulators as if they are in the microgravity environment. Therefore, the accuracy of the experiment is only influenced by the accuracy of the industrial robots and the accuracy of the dynamic model of the spacecraft, robotics and space environment [69].

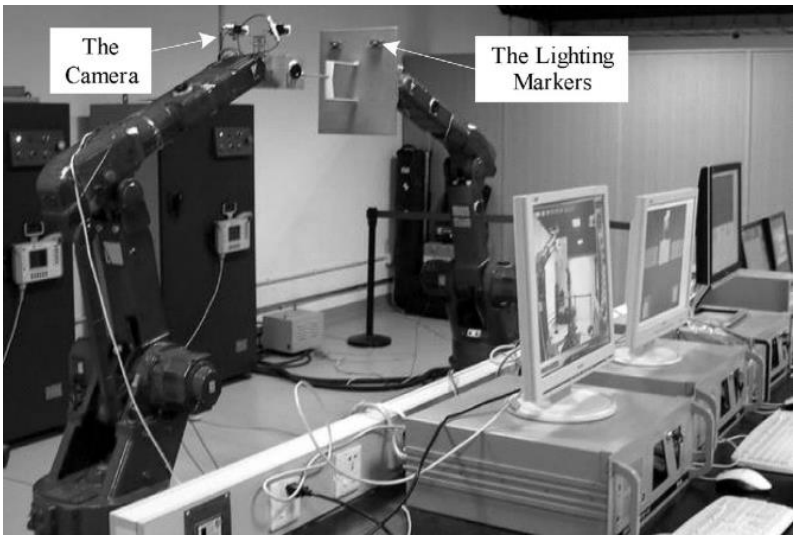


Figure 2.2 Industrial robotic manipulators [49].

Drop towers and parabolic flight offer equivalency to the actual space environment by allowing free-flying/floating in 6-DOF. The residual gravity from a drop tower is small offering adequate equivalency to the microgravity environment. The residual gravity from a parabolic flight is larger and imposes more disturbances on the free-flying object. The benefit of the parabolic flight is the repeated, and longer test periods. The duration of these test is limited (10s for drop tower and 25s for a parabolic flight). They both have very limited available testing space where the test can be performed [69].

Due to all these limitations, the planar air-bearing microgravity simulator provides the most cost-effective testbed for the validation of space robotics and the equivalency to the microgravity environment, see Figure 2.3. A satellite with a robotic manipulator can move and rotate freely but is limited to the 2D planner motion [69].

Figure 2.3 Planar air-bearing microgravity simulator [69].

2.2 Air-Bearing Simulators with Robotic Manipulators

As this thesis focuses on the dynamics and control of small spacecraft with robotic manipulator, similar air-bearing satellite-manipulator testbeds will be reviewed here.

The testing system developed by the Space Research Centre of the Polish Academy of Sciences consists of a 2-DOF manipulator mounted on a base satellite, see Figure 2.4. The total length of the manipulator is 1.22 m, and the total mass of the simulator is 18.9 kg. Each joint consists of a DC motor and an optical encoder. The simulator consists of an On-Board Computer (OBC) and two joint-control boards (JC). The OBC is a 1GHz DM3730 Texas Instruments processor and the JC are 32-bit ARM Cortex M3 microcontroller with accompanying electronics to communicate and power the motor and encoders. Visual markers are placed on the base simulator and the links of the manipulator

to determine their positions using external cameras and of-board computers. The pose information is then transmitted to the base simulator using Bluetooth. Data is sampled and logged at a rate of 100 Hz. The simulator's OBC computes all path-planning and control. The simulator sits on a 2m x 3m granite table [71].

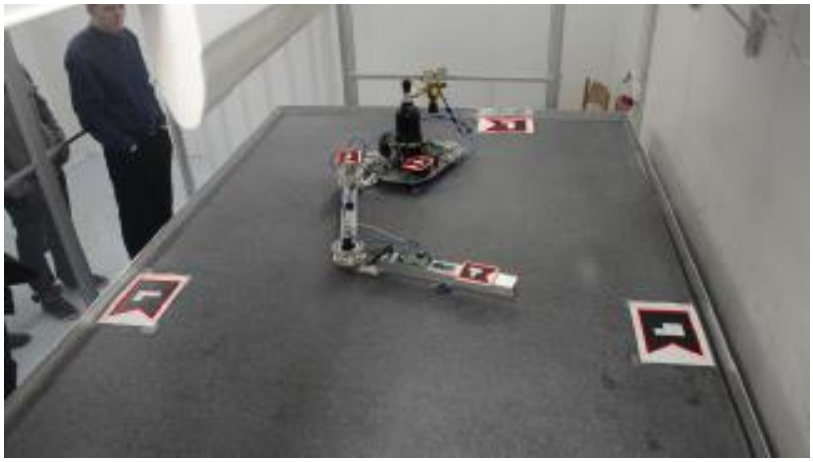


Figure 2.4 The Polish Academy of Sciences simulator [71].

The US Naval Postgraduate School developed its own Floating Spacecraft Simulator Test Bed (POSEIDYN), see Figure 2.5. Its 4th generation air-bearing simulator system consists of a 10 kg target simulator and a 13 kg chaser, both with 3 DOF of motion. One simulator has a 4 DOF manipulator with 8 cold-gas thrusters to provide translational and rotation motion. The chaser also has a speed-controlled reaction wheel to provide fine rotational motion control [73]. Each link in the manipulator consisted of a 1.8 Nm motor with an encoder, torque sensor and Wi-Fi module to transmit data back to the base simulator. The total mass of the manipulator was the same as the base satellite. An overhead

optical motion capture system (VICON) is used to track the position and orientation of the simulators. The VICON system then transmits the data to the simulators. The joint and simulator position data are recorded and updated at a rate of 50 Hz [73]. The onboard computer of the system is an Intel Atom 1.6 GHz running Ubuntu 10.04 32-bit server with PREEMPT-RT installed to make the operating system real-time [74]. Navigation and control are computed on the simulators. The simulator sits on a 4m x 4m granite table [73].

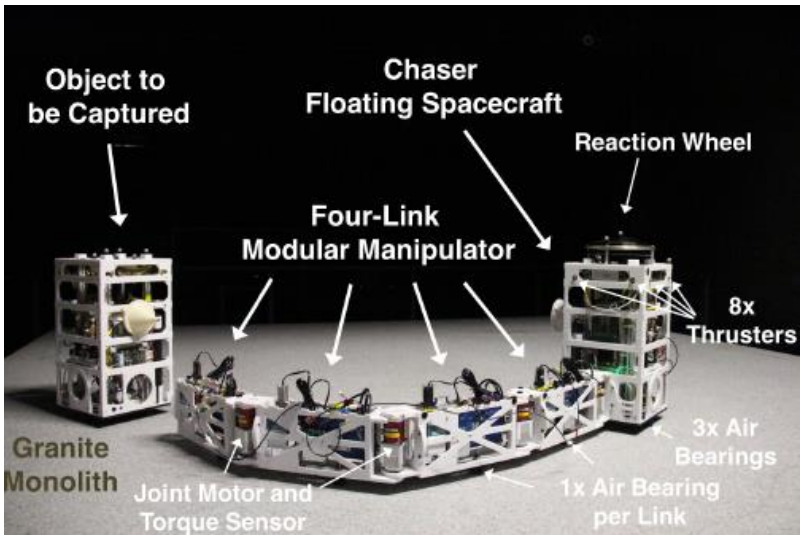


Figure 2.5 POSEIDYN satellite simulator [73].

The PINOCCHIO (Platform Integrating Navigation and Orbital Control Capabilities Hosting Intelligence Onboard) testbed designed by the Guidance and Navigation Lab at La Sapienza – University of Rome consists [75] of a 10 kg simulator, with an attached 3 DOF robotic manipulator, see Figure 2.6. The manipulator is actuated with three stepper motors with a 0.1 Nm torque. The base simulator has eight on-off gas

thrusters. The OBC consists of a 1.6 GHz Intel Atom processor connected to inertial measurement unit and actuators to perform trajectory planning and control. An external camera measures the position and orientation of the simulator and relays this information to the OBC on the simulator [76].

Figure 2.6 PINOCCHIO air-bearing simulator [76].

At the National Technical University of Athens (NTUA), a dual arm air-bearing simulator has been developed with a total weight of 14 kg, see Figure 2.7. The granite table has a surface area of 2.2m x 1.8m. An overhead camera is used to track the robot's position and orientation by tracking three LED's mounted on the robot where the images are

processed off-board and then the position and orientation information is transmitted through Wi-Fi to the OBC on the simulator. The OBC is an Intel Celeron 648 MHz to process the navigation and control. Each link of the manipulator is actuated with a DC servomotor with the addition of a force sensing gripper [77].

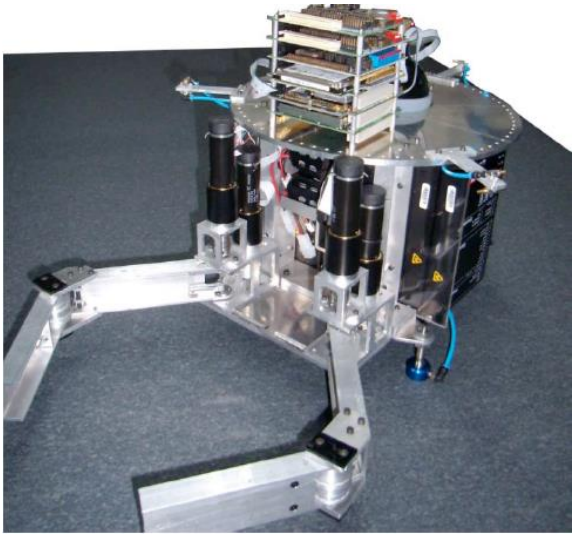


Figure 2.7 NTUA air-bearing planar simulator robot [77,78].

Carlton University developed an air-bearing simulator system to test different gripping capture mechanism and different control techniques, see Figure 2.8. The base satellite simulator called SPOT is a 30 cm x 30 cm x 30 cm cube and uses air-bearings to create a microgravity environment. It possesses a Raspberry Pi 3 as the OBC running Raspbian Jessie Linux. An external system is used to capture the position and orientation of the satellite simulator and the robotic joints which can measure at a rate of 240 Hz with

an accuracy of 0.01 mm. No onboard optical sensors are used to determine the position and orientation of the satellite simulator, nor the joint angles of the robotic manipulator [79].

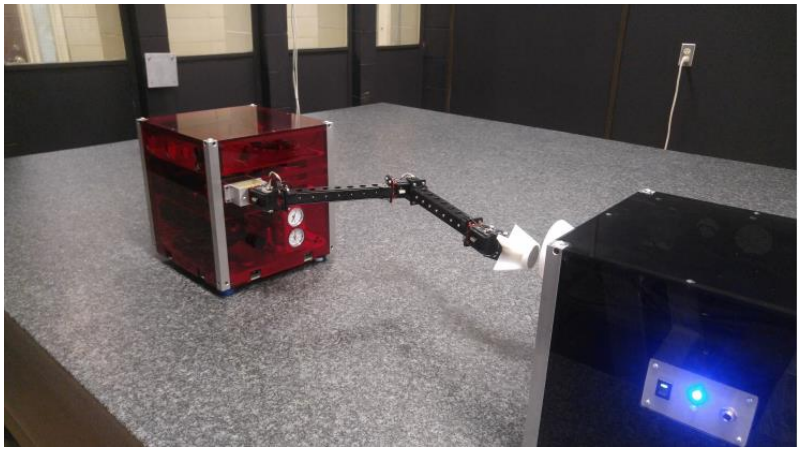


Figure 2.8 SPOT facility with two air-bearing simulators [79].

In summary, existing testbed suffer from two limitations. The first, being that the testbeds use external system when determining the simulators own position. These external systems do not limit their processing, and therefore do not mimic the computational limitations expected by on-board computers. Using these external systems also suffer from a limitation due to the wireless transmission of data. Transmission over a wireless UDP connection leads to inevitable dropped measurements due to lack of error correction [74], which is undesirable. Other protocols provided undesired latency [74]. Using an on-board computer limits the processing capabilities to that expected from OOS spacecraft and removes the limitations of wireless data transmission.

The chaser spacecraft must compute their own position in orbit and these external systems removes this computational requirement. This computational limitation needs to be considered when developing for OOS. Using the previously developed positioning system developed for the on-board computer, adds computational limitations on the other algorithms developed for the system. Doing this makes this testbed the first developed air-bearing simulator with a robotic manipulator using an on-board positioning system.

The second limitation is that the same external systems are used to determine the position of the target. Servicing vehicles would be equipped with cameras and other sensors to determine the position and orientation of the target. This imposes additional load on the on-board computers. These external systems update at a rate of 50-1000 Hz [73,80] while update rates of 4-10 Hz are expected [49,81]. A vision system must be developed for the testbed that mimics the expected criteria for OOS vision systems and is implemented in this work. These vision systems must be developed to function on the limited hardware and in tandem with the on-board positioning system which imposes additional computational limitations.

Chapter 3 DEVELOPMENT OF EXPERIMENTAL FACILITY

Summary: The Development of Experimental Facility chapter contains three sections. The first section covers the test testbed overview and development. This includes the existing equipment developed repaired and upgraded from the existing facility. The second section includes the instruments used and developed to determine the position and orientation of both the chaser and target simulators. The final section is the software developed to control the paths of the robotic manipulator, actuate the gripper, and finally execute the path planning of the chaser simulator.

3.1 Testbed Overview

The air-bearing testbed in the Space Engineering Design Lab at Lassonde School of Engineering consists of a 2m x 4m x 0.5m granite slab produced by Standridge Granite. The slab sits on a 3-point supports on a steel frame with four legs. The four legs are adjustable to level the table [82]. The current table is settled to 0.2 deg inclination of the top surface of the table. This results in an acceleration of 3.08mm/s^2 in the plane. The table level cannot be easily corrected as one of the legs is damaged. In addition, the

lab does not own a level with high enough accuracy to appropriately level the table. The thrusters of the simulator can overcome this acceleration therefore leveling was not required.

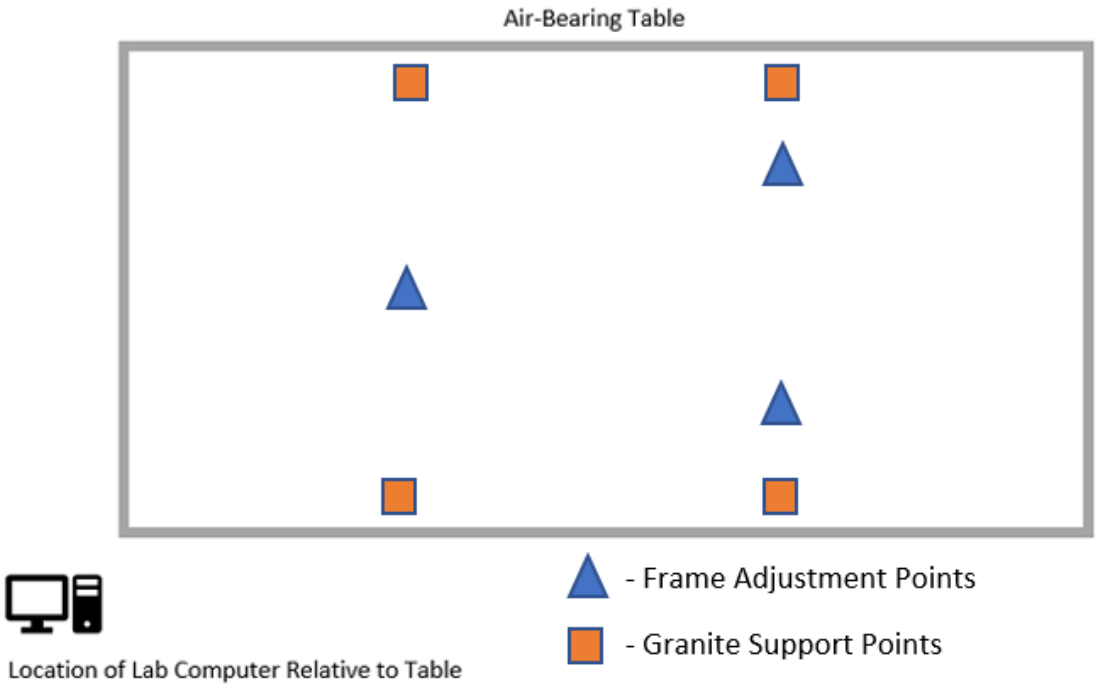


Figure 3.1 Granite table support and adjustment points.

There is a vast difference between the dynamics of a ground manipulator and a space-based manipulator as the space-based manipulator is not anchored to the ground. This will result in the manipulator being dynamically coupled to the base spacecraft, which disturbs the pose of the base spacecraft in reaction to the manipulators motion [83]. Therefore, “the control system of the satellite should either be able to fully compensate reaction torques and reaction forces induced by the manipulator motion or the system

should be switched-off during the final phase of the capture maneuver. In the latter case the system is in free-floating state and influence of the manipulator motion on the state of the satellite must be taken into account by the control system of the manipulator" [69,84].

In the case of our testbed, the accelerations induced by the level of the table are too great to test the free-floating case of the chaser simulator. In addition, the simulator is not placed in a clean room environment and therefore suffers from dust particles on the surface of the table, even with the surface cleaned frequently before experiments. These act as disturbances to the simulator that would not be present in the space environment. Therefore, the control system will compensate for the reaction torques and forces from the environment and the attached robotic manipulator using the thrusters on the base simulator.

3.1.1 Air-Bearing Robotic Simulator

The original development of the satellite simulator testbed was constructed by Tsinghua University [85] and was further developed by visiting professor Ning Chen, Sat Li for his PhD and Joshua Cookson for his master's degree [82]. The satellite simulators operate in the plane with 3-DOF - two translational DOF and one rotational DOF. The satellite simulators float on air providing a near frictionless surface and simulating the microgravity environment.

Each simulator contains two high pressure air tanks with a total capacity of 4.0 L with a maximum pressure of 20 MPa. The tanks are connected to a pressure regulator that reduces the pressure from 20 MPa to 1MPa which is then connected to another regulator that reduces the pressure further to 0.4 MPa. The 0.4 MPa pressure lines are then connected

to three air bearing feet and 8-unidirectional gas thrusters. The gas thrusters then output a constant force of 0.065N from each thruster [17]. Figure 3.2 and Figure 3.3 show the chaser and target simulators on the granite table.

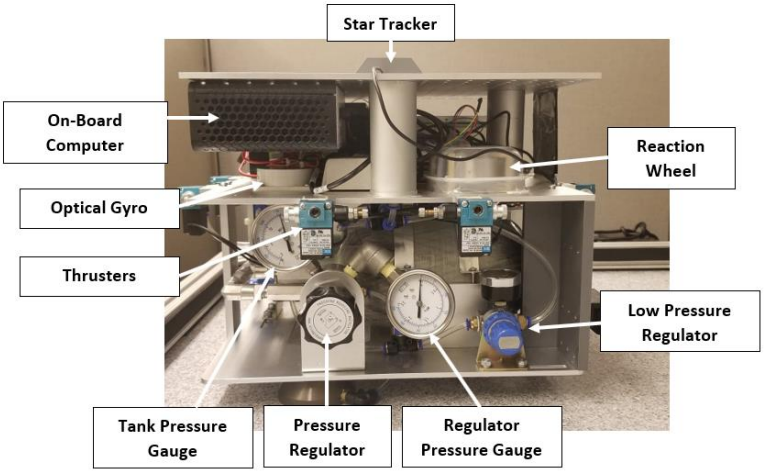


Figure 3.2 Components of the base simulator.

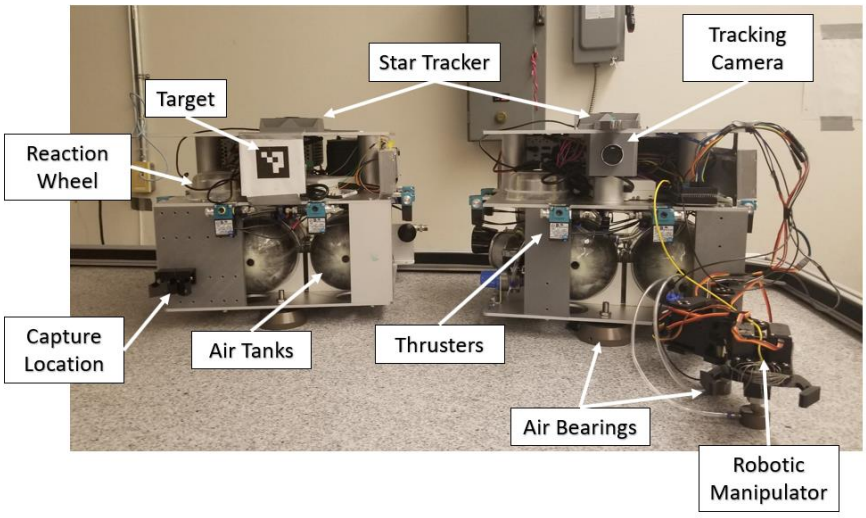


Figure 3.3 Target and chaser simulators.

An onboard 12V LiFePO4 battery powers all the onboard components. An onboard power distribution rack is used to convert the power to 5V, 12V and 24V. A separate converter is used to step up 12V to constant 19V for the onboard computer.

The onboard fanless computer consists of an i7-8550U processor with 16 GB of RAM and a solid-state drive running Windows 10. The onboard computer is passively cooled as the spinning fan will provide unwanted momentum and force to the base. If a conventional hard-disc drive were used, unwanted momentum would also be induced onto the base satellite. The system runs Windows 10, as legacy code for the simulators were developed in LabVIEW, which can only run on the Windows Operating System. Using LabVIEW also allows for the rapid development and testing of control algorithms.

The simulators are also equipped with onboard reaction wheels, which spin to rotate the satellite, through the conservation of momentum.

3.1.2 Robotic Manipulator Assembly and Integration

The movement of the robotic manipulator imposes disturbances on the servicing satellite. How much the movement imposes depends on the mass and moments in comparison to the chaser platform. When the platform is significantly more massive than that of the manipulator these disturbances are nearly negligible. This is the case for the Canadarm and Canadarm2 on the Space Shuttle and the ISS [76]. OOS missions are expected to have the chaser satellite platform and the manipulator properties to have equal or similar mass ratios [76,86]. When developing testbeds creating dynamically equivalent systems is important so that it can representative of full-scale systems [86].

Before designing the robotic manipulator, an in-depth review of space free-flying/floating robotics for past, planned, cancelled and future missions was investigated to determine what parameters should be used for the design of the robotic manipulator. Only missions where the mass of the manipulator and the base satellite were available are included. Among the included missions, the ratio of the manipulator length to the base spacecraft length ranged from 0.769 – 1.7956, while the mass ratios of the manipulator to the base spacecraft were 0.0187 - 0.0829. This provided the guidance for the design of the manipulator for the air-bearing testbed.

In addition to the above guidance, the manipulator was designed to accommodate the existing testbed in the lab for the scenario where the target simulator is placed in the middle of the table and would require more than a 90^{o} rotation to detumble the target simulator. The coupled simulators would traverse through the short side of the table and thus must not interfere with the railing, see Figure 3.4.

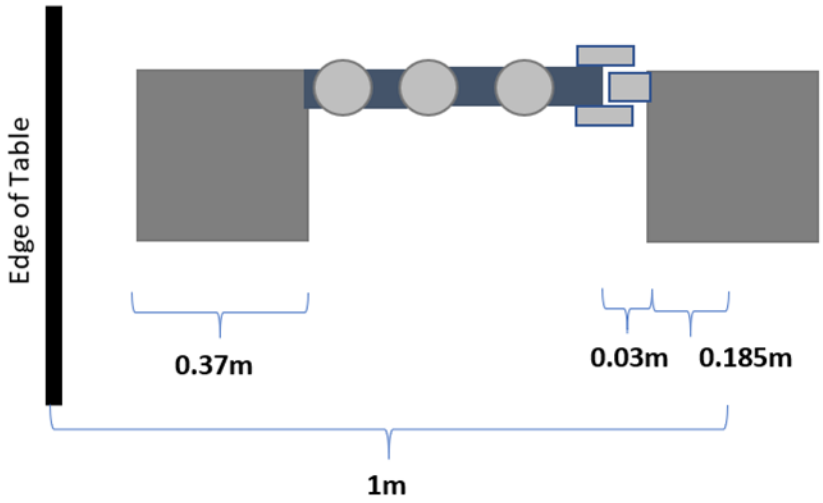


Figure 3.4 Length of manipulator using table constraints.

$$1m - 0.37m - 0.185m - 0.05 = 0.394m \tag{3.1}$$

Therefore, the manipulator must have a total length less than 0.394m when the manipulator is fully extended and has captured the target simulator. Accordingly, the manipulator was designed with a length of 0.355m leaving 0.039m of additional space for the manipulator to avoid the railing on the table edges.

When the manipulator is in its starting position, the total length of the manipulator is 0.297m. This starting position length is due to the limitation of the servomotors. There is a limited region in which the servomotors can accurately operate, and this starting region is at the end limit that the motors can reach in the compact position. The manipulator has a total mass of 597g allowing it to fall within the mass ratio guidance.

Table 3.1 Designed manipulator mass and length

Manipulator Feature	Goal criteria [% of base length]	Simulator	Manipulator	% Ratio
Length	76.9 - 179.56	0.37m	0.355m	95.9
Mass	1.87 - 08.23	18.0 kg	0.597kg	3.32

3.1.3 Manipulator and Gripper Assembly

The manipulator has two air-bearings to support its weight and to keep the manipulator level. In the original design only a single air bearing was installed. Due to the table not being in a clean room environment the bearing would get stuck by dust on the granite table and would prevent the simulator from moving. Adding the second bearing to the manipulator provided enough support to prevent the system from getting stuck, see Figure 3.5.

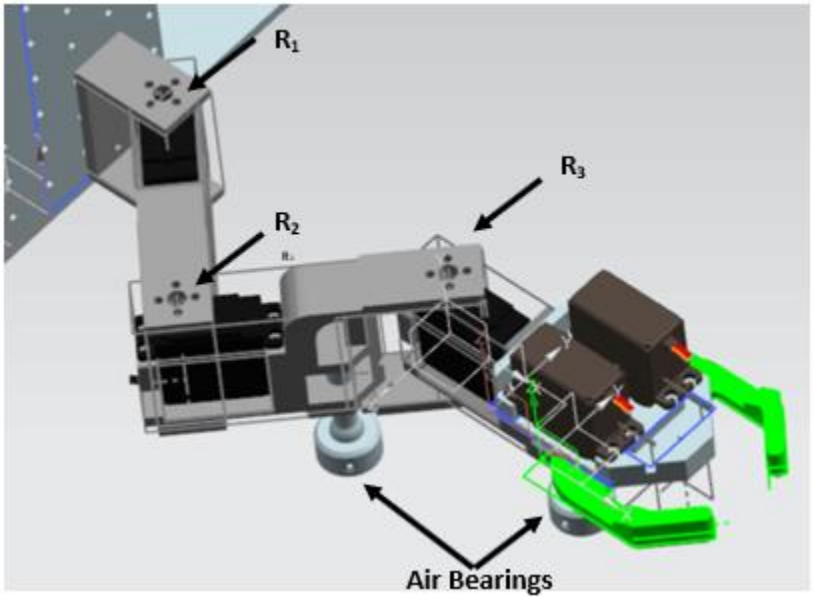


Figure 3.5 CAD design of 3-DOF manipulator.

The gripper consists of two fingers with a triangular shaped structure. The triangular structure guides the manipulator to the target where it can soft dock with the target by making contact in the cone like region. Then gripper fingers can then reach around the capture point and hard dock with the target's capture point making a secure connection.

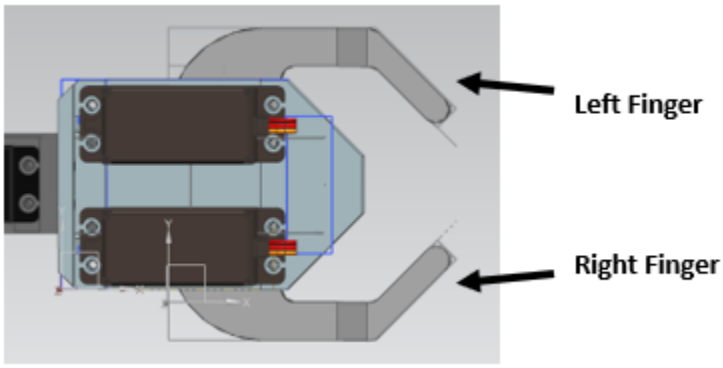


Figure 3.6 CAD design of gripper.

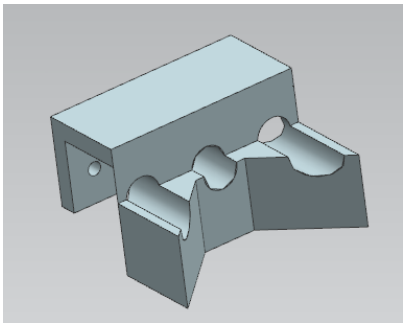


Figure 3.7 CAD design of target capture fixture.

The servo motors used for the manipulator were DS3218 and MG996R servos, made by Dsservo and Towerpro respectfully. The corresponding stall torques for the two servos are 1.77 Nm and 0.92 Nm, respectively. The DS3218 has an operational range of 180^{o}, while the MG996R has an operational range of 120^{o}. The DS3218 servos were used for the first and third joints and the MG996R for the second and gripper fingers. The MG996R link was chosen for the second link as this link requires the least travel distance, see Figure 3.5. Using this limited motion servo on the second joint also prevents the manipulator from experiencing kinematic singularities.

As this was the first implementation of a robotic manipulator on this platform, the goal was to choose a low-cost manipulator that was a complete off the shelf unit for quick integration. A full manipulator was purchased for $130.00, however the rigidity of the links was too low and was easily bent. The manipulator was re-designed using 3D printed parts to increase its rigidity and allow for the easy mounting of the air-bearings. To keep costs low the servo motors from the original manipulator were re-purposed.

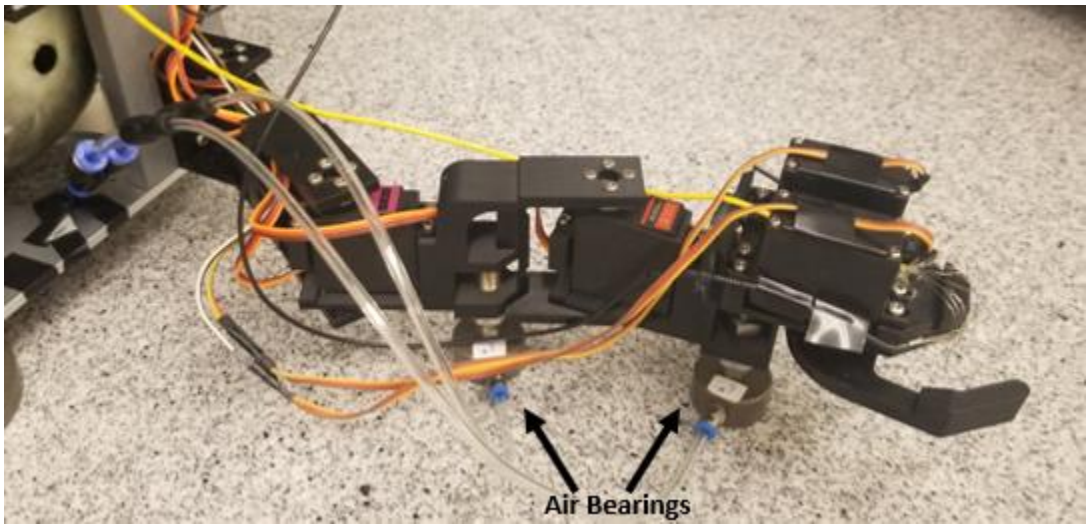


Figure 3.8 The assembled real manipulator.

3.1.4 Manipulator and Gripper Actuation

The manipulator has five servo motors. Three of them provide the planar motion and two actuate the gripper. Servo motors were chosen for their inexpensive cost and simple control, being a good first implementation for the simulator. Servo motors have a built-in P-controller that makes them simple to control by inputting a specified PWM signal to a corresponding angle. This simple implementation allows for a relatively accurate open-

loop control for a limited region making it relatively simple to implement. The drawback of using these servo motors is that they do not provide feedback of the motor positions to the OBC, and therefore more robust path planning and control cannot be implemented. A 5 DOF robotic kit was purchased for $130 and was modified to accommodate the simulator structure.

In the original design, a single motor was used to drive both gripper fingers. The left finger was driven by the servo motor and a gear drove the right finger, see Figure 3.6. However, as the gear was 3D printed it was prone to wear and slippage making the system fail repeatedly. It was decided to place a servo motor for each gripper to prevent this slippage problem.

Below is a schematic of the manipulator electrical system as shown in Figure 3.9. An Arduino microcontroller with an attached Robotics Control Shield for Arduino Mega 2560 R3 was used to increase the power provided to the servo motors. To communicate between the Arduino controller and the on-board computer running LabView, the LabView Arduino Interface is loaded onto the Arduino and installed to LabView.

The shield is connected to a voltage step down converter which is then connected to the battery power supply. The shield was set to 5V for the attached motors. The power provided by the battery can range from 12V – 14V. A voltage step down converter is installed between the battery and the robotic control shield that regulates the voltage to 12V.

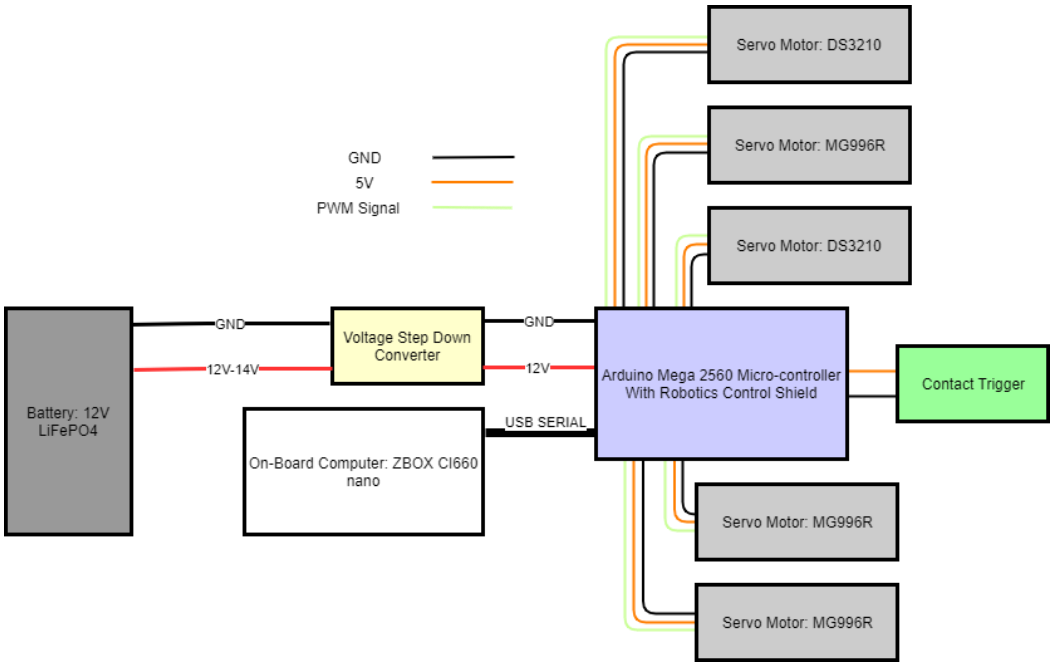


Figure 3.9 Electronic diagram for robotic manipulator.

There are two ways for the system to initiate capture. The first is when the system's forward kinematics have the end-effector aligned with the target capture point. The second as a backup, in the event the end-effector does not align appropriately due to the error from the servo motors, is the "frayed wire contact." Different trigger methods were implemented and tested such as using pushbuttons and lever switches. The force required to close the switches was too large, even after modification. The frayed wire design required the lowest force while still falling within the budget constraints.

On the end of the manipulator, frayed wires are spread overhanging in front of the cone soft docking point connected to an analog input pin on the Arduino Mega shield. A separate frayed line connected to 5V power runs horizontally across the cone assembly.

When the manipulator contacts the capture point, the frayed analogue input wires will contact the 5V power line closing the circuit. When the circuit is closed, the Arduino microcontroller will read the voltage and command the manipulator to close its end-effector to capture the target.

Table 3.2 Mapping of Arduino pins to the corresponding manipulator joints

Arduino Pin	**Servo Motor**
Digital PWM 8	Joint 1
Digital PWM 9	Joint 2
Digital PWM 10	Joint 3
Digital PWM 11	Left Gripper
Digital PWM 12	Right Gripper
Analog 0	Contact Sensor

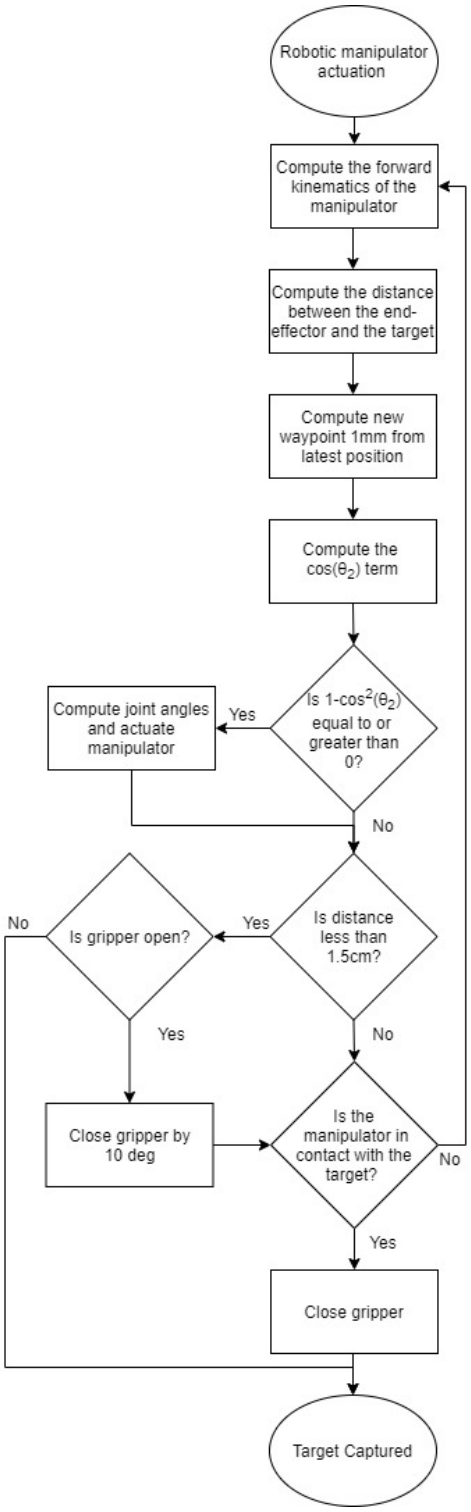


Figure 3.10 Robotic manipulator path planning and gripper actuation.

3.2 Instrumentation and Measurement

3.2.1 Star Tracking Camera

Mounted on the top of the simulators are Logitech C920 USB cameras fitted with infrared (IR) bandpass filter. Mounted on the ceiling are IR LED lights that are captured by the camera and fed into onboard computer to calculate the location and orientation of the simulator in the world frame fixed to the granite table. The LED map was designed so that at any instance at least five points would be visible in the camera's field of view [82]. This was developed by Joshua Cookson, a master's student in the Earth, Space Science and Engineering department, that focused his research on autonomous docking and rendezvous of spacecraft [82].

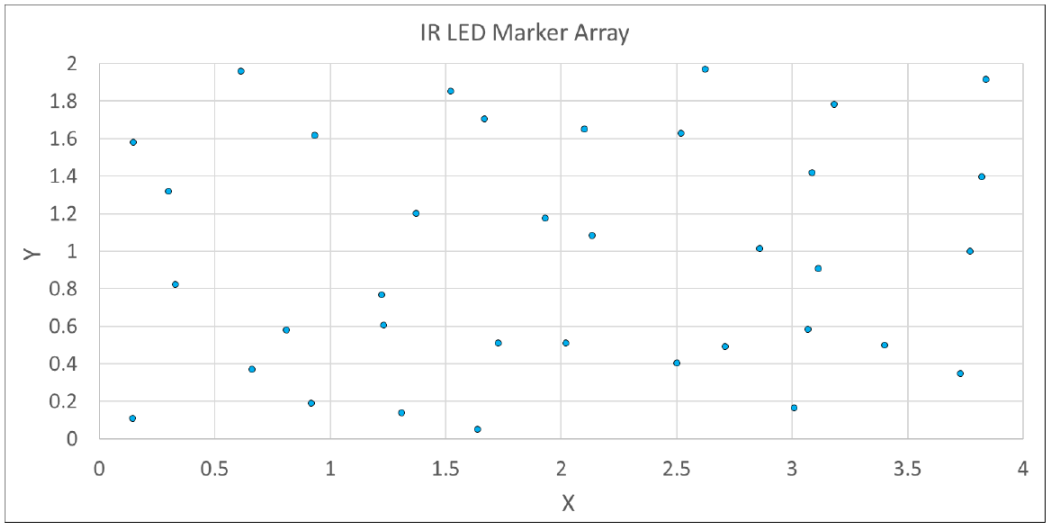


Figure 3.11 IR LED star map positions with respect to the table [82].

Figure 3.12 Installed LED star map [82].

Figure 3.13 Stars viewed by the camera on simulator through IR filter [82] (enhanced).

To provide the position and attitude information, the simulator identifies the stars using a stored catalog that has the geometric relationships between the stars. For any given star the four closest neighboring stars are stored along with its geometric relationship between them [82]. By using the star tracking system, the simulator can determine its own position and orientation without relying on external sensors and computers to compute its pose information. The pose of the system is updated at a rate of 9 Hz. The position data has a 3-point moving average filter to reduce the level of noise in its position. The star tracking system makes the simulators truly autonomous.

The star tracking system could periodically result in large jump errors, which causes the position to deviate as much as 1m. This is due to the simulator identifying noise as a star when the number of LEDs in the frame of the camera falls below the required five to compute the position and orientation. To filter through these errors, a check between the last recorded position data and the current position data has been added. This determines if a spike occurred in the measured velocity simulator.

To determine the threshold for checking the jump velocity, we must determine the maximum acceptable velocity that the base simulator can move. From other literature, the maximum recommended velocity of the end effector of space robotic manipulator is 40 mm/s [49]. The end effector's velocity is affected by the robotic manipulator and the base satellite. Therefore, the maximum velocity that can be induced by the base and manipulator must be set. The way pointing for the manipulator was set small (0.5mm/loop) to produce smooth motion. This relates to the minimum distance at which the final motor can actuate

the end-effector. This results in a maximum velocity of the manipulator's end-effector to be 3.57 mm/s in experiment based on the following calculation.

$$0.50mm/s \div 0.14s/loop = 3.57mm/s$$

Therefore, the maximum velocity of the base simulator is 36.43 mm/s. For each loop iteration this produces a maximum velocity of 4 mm/loop.

$$40mm/s - 3.57mm/s = 36.43mm/s$$

$$36.43mm/s * 0.11s/loop = 4.01mm/loop$$

If the next measurement exceeds this value, the last position value is used and the value with the suspicious jump is discarded.

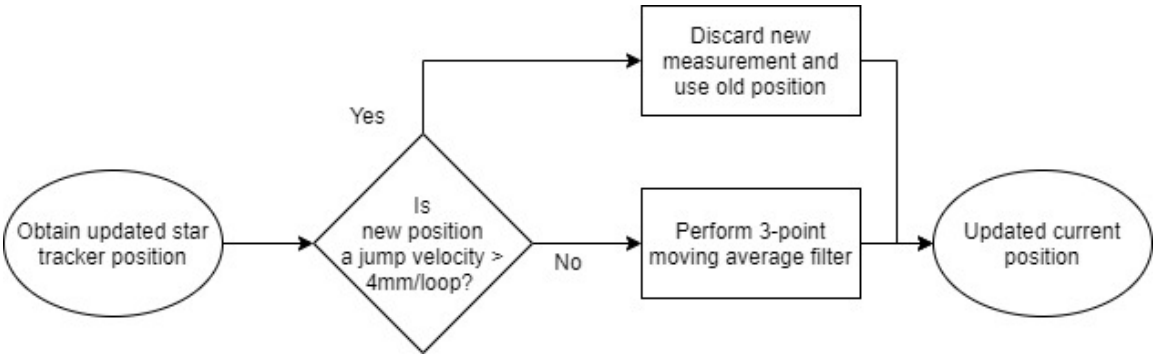


Figure 3.14 Star map error correction.

3.2.2 Target Tracking Camera

To track the target satellite, a camera is placed on the side of the simulator that represents the chaser satellite. The tracking module was written in C++ using OpenCV and the Aruco Library. An Aruco marker was placed on the target satellite, see Figure 3.15.

The Aruco marker makes use of the known size of the marker and the 4 corners to determine the position and orientation by a single camera. For a full explanation of how the Aruco Library works please see the paper "*Automatic generation and detection of highly reliable fiducial markers under occlusion*" [87]. Using the inner pattern, the unique ID of each marker can be determined. In the future, a marker can be placed on each face of the target satellite simulator to determine its orientation when not facing the capture face. For the purpose of this thesis, only a marker on the target face is required to demonstrate the approach and capture of the target simulator.

Figure 3.15 Sample Aruco marker.

To use the module the camera parameters must be determined for OpenCV. The camera used is a Microsoft LifeCam Studio. This camera was chosen as it was a spare camera in the lab and was no cost. As the camera parameters were not know they were determined using the Camera Calibration Toolbox for Matlab [88], using a checkerboard pattern. This toolbox is also implemented in in OpenCV however, the Matlab version was

chosen for its easy to use graphical user interface. The toolbox uses the known pattern and several pictures to determine the parameters for the camera.

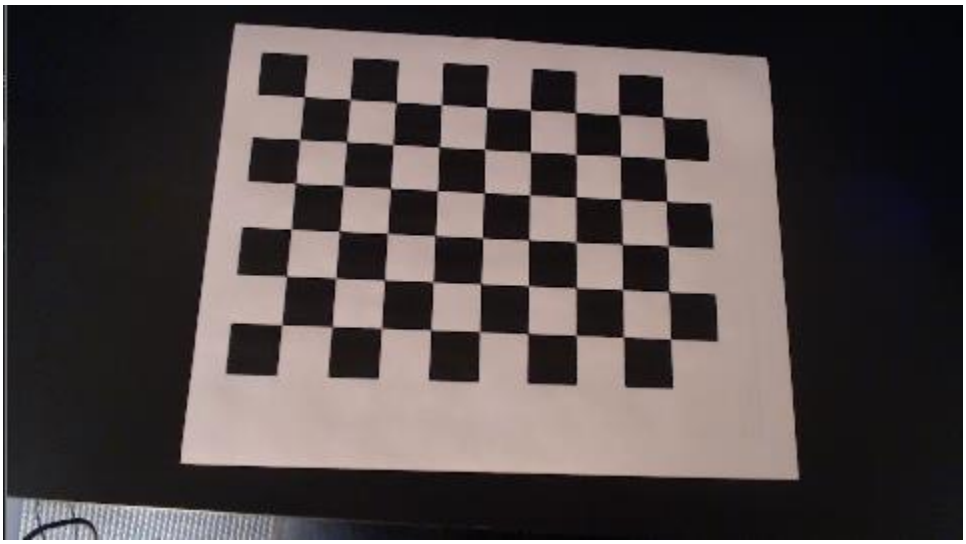


Figure 3.16 Sample image used during the calibration process.

To send the camera information to the main LabView code, the Cesanta Mongoose Embedded Web Server [89] is used. When the main LabView code requires the camera information, LabView will send an HTTP/GET request to the Cesanta Mongoose Embedded Web Server, see Figure 3.17. This triggers the C++ module to take the latest image the camera has captured and process its pose information using the Aruco Library. It then returns the pose information and target ID in an XML file. The XML file is structured so it can be read as a LabView cluster.

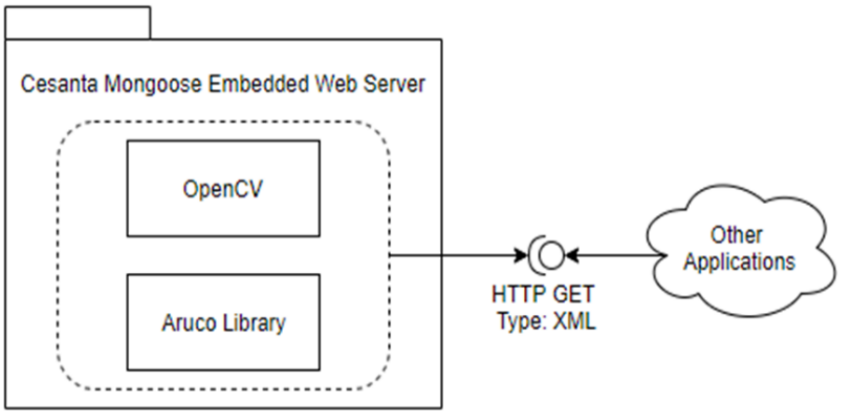


Figure 3.17 Tracking software sub-components and interface.

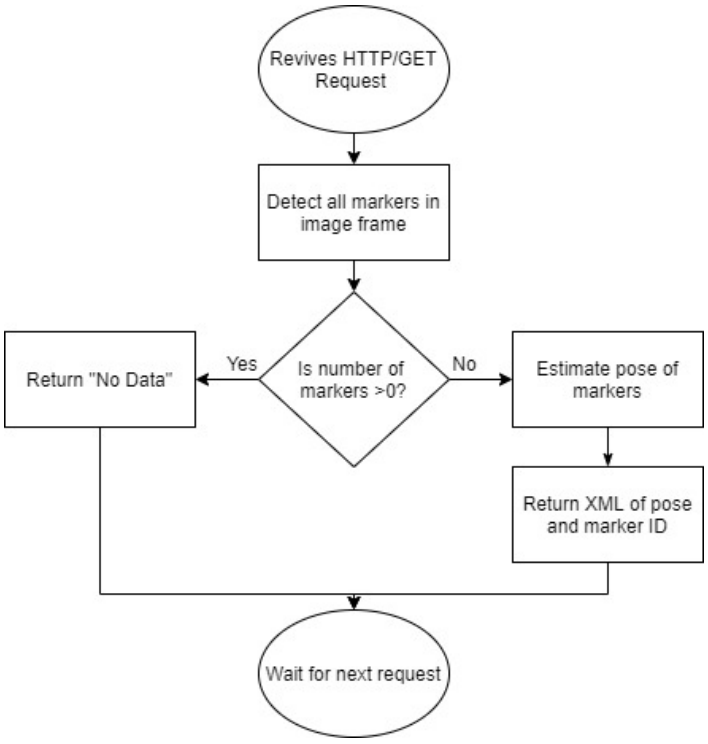


Figure 3.18 Tracking software trigger, flow and return data.

Highlighted in red in the below XML is where the computed values for the attitude, X and Y distances in the camera frame are returned.

```
<Cluster>
<Name></Name>
<NumElts>4</NumElts>
<DBL>
        <Name>ID</Name>
        <Val>ARUCO_ID</Val>
</DBL>
<DBL>
        <Name>ATTITUDE</Name>
        <Val>ATTITUDE_DEG</Val>
</DBL>
<DBL>
        <Name>X</Name>
        <Val>X_DISTANCE_M</Val>
</DBL>
<DBL>
        <Name>Y</Name>
        <Val>Y_DISTANCE_M</Val>
</DBL>
</Cluster>
```

To determine the pose of the target simulator in the world frame, the pose information of the Aruco marker is first measured. Using the known position of the marker on the target simulator, the pose of the target can be determined. Using the same transformation, the position and orientation of the capture point can be determined.

In the camera frame, the X position of the target its horizontal position left or right from the camera. The Y position is the distance perpendicular to the camera plane. The attitude of the target simulator is the angle between the vector perpendicular to the camera frame and the vector perpendicular to the Aruco marker in the counter-clockwise direction (in accordance with the right-hand rule), see Figure 3.19.

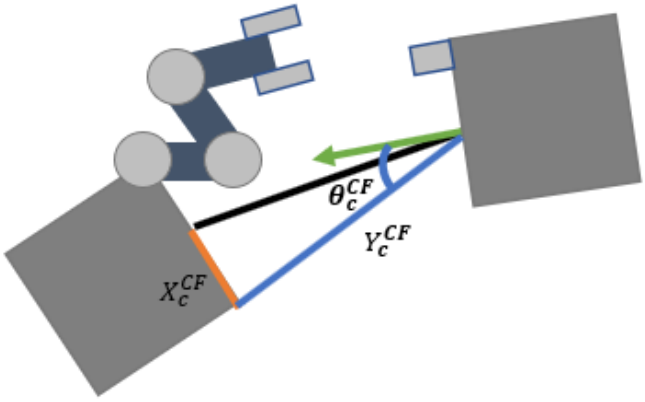


Figure 3.19 Position and orientation of the Aruco marker.

To convert the camera data in the camera frame to the target's position and orientation in the world frame, the target simulator position and orientation must first be determined in the body fixed frame of the chaser simulator, see Figure 3.20. The body dimensions of both the target and chaser simulators are both 0.37m x 0.37m.

$$\begin{bmatrix} X_{TS}^{BF} \\ Y_{TS}^{BF} \\ \theta_{TS}^{BF} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c^{CF} \\ Y_c^{CF} \\ \theta_c^{CF} \end{bmatrix} + \begin{bmatrix} T_s + T_s \cos(-\theta_c^{CF}) \\ T_s \sin(-\theta_c^{CF}) \\ 0 \end{bmatrix} \tag{3.2}$$

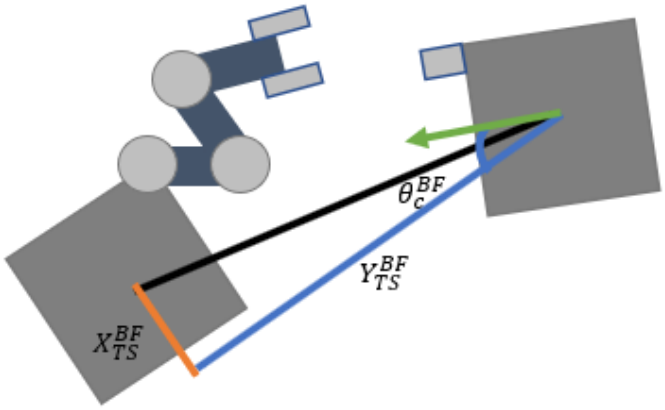


Figure 3.20 Position and orientation of the target simulator with respect to the chaser.

The target in the body fixed frame is then translated into the world frame by rotating the body fixed frame. It is then translated by the chaser simulator's position. The orientation of target simulator is determined as the orientation of the target simulator in the body fixed frame plus the orientation of the chaser simulator in the world frame plus an orientation conversion of π as the Aruco marker is facing the chaser simulator, see Figure 3.21.

$$\begin{bmatrix} X_{TS}^{WF} \\ Y_{TS}^{WF} \end{bmatrix} = \begin{bmatrix} \cos(\theta_{CS}^{WF}) & -\sin(\theta_{CS}^{WF}) \\ \sin(\theta_{CS}^{WF}) & \cos(\theta_{CS}^{WF}) \end{bmatrix} \begin{bmatrix} X_{TS}^{BF} \\ Y_{TS}^{BF} \end{bmatrix} + \begin{bmatrix} X_{CS}^{WF} \\ Y_{CS}^{WF} \end{bmatrix} \tag{3.3}$$

$$\theta_{TS}^{WF} = \theta_{TS}^{BF} + \theta_{CS}^{WF} + \pi \tag{3.4}$$

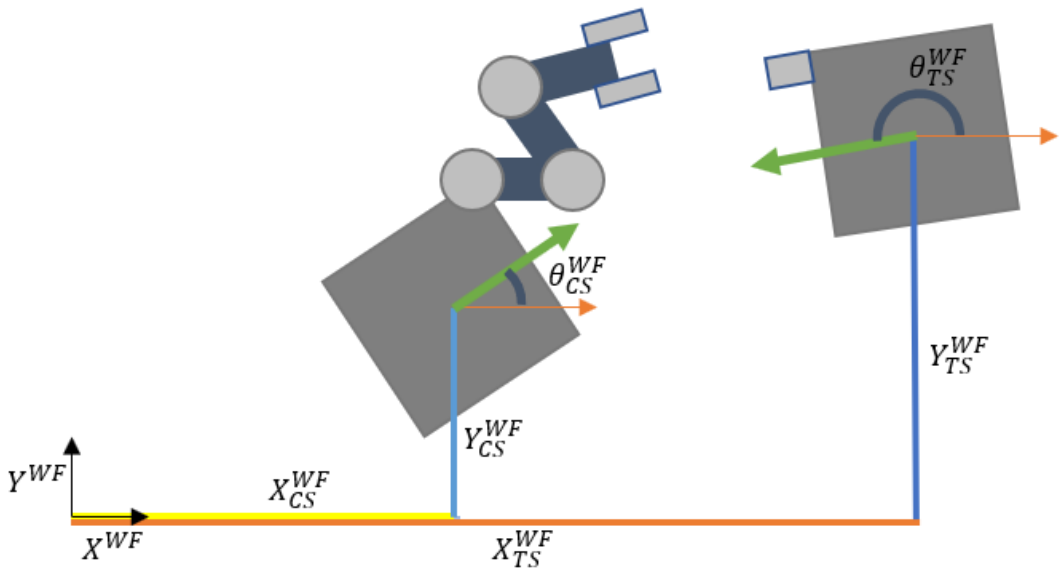


Figure 3.21 Position and orientation of simulators in world frame.

Due to the length of the robotic manipulator and the limitation of its joint angles, the closest distance the chaser satellite simulator can approach the target is 0.75m from the center of each simulator. In the camera frame, the shortest distance from the Aruco marker is 0.35m to the camera. At this capture distance, the pose data of the Aruco marker is recorded and displayed. The reason to choose this sample distance is at this distance the chaser will attempt to capture the target and the sampled results are most crucial to this task. At any distance beyond this value, the noise will increase but will reduce to that of the sampled results as the chaser moves towards the target.

The tolerance for the manipulator to capture the target was 1.25 cm and an orientation offset of 0.05 rad (2.86 deg). Here, 1.25 cm is the chosen waypoint switching distance that worked well with the controller as it exceeded the measured noise from the

star-tracking camera. 0.05 rad (2.86 deg) was also used as it exceeds the measured noise from the star tracking camera. Due to the limitations of the open-loop manipulator, the feed-forward control had significant error in the orientation of the manipulator beyond ±0.1 rad from being perpendicular to the chaser simulator, see Figure 3.22. Half this value was chosen to be the capture requirement for the manipulator workspace to ensure that a failed capture would not be caused by the misalignment of the manipulator.

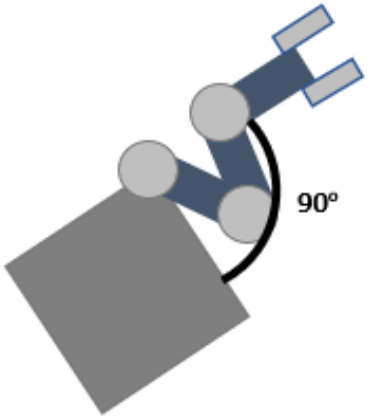


Figure 3.22 Starting Orientation of the manipulator end effector.

The noise from the raw data caused the measurements to fall outside of the tolerable capture requirements for the y distance and the orientation in the camera frame. A 3-point moving average was implemented to reduce the noise to an acceptable level to meet the capture requirements, see Figure 3.23-Figure 3.25. A moving average with more than 3 points could further reduce the noise but caused the system to lag beyond what can be corrected by the path planning algorithm in the tumbling case.

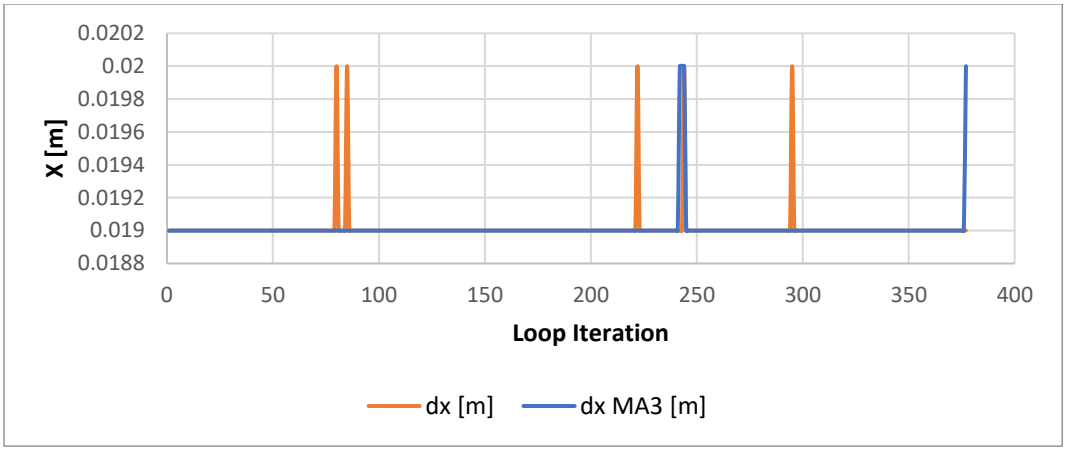


Figure 3.23 Raw data and 3-point moving average filter of X position data in the camera frame.

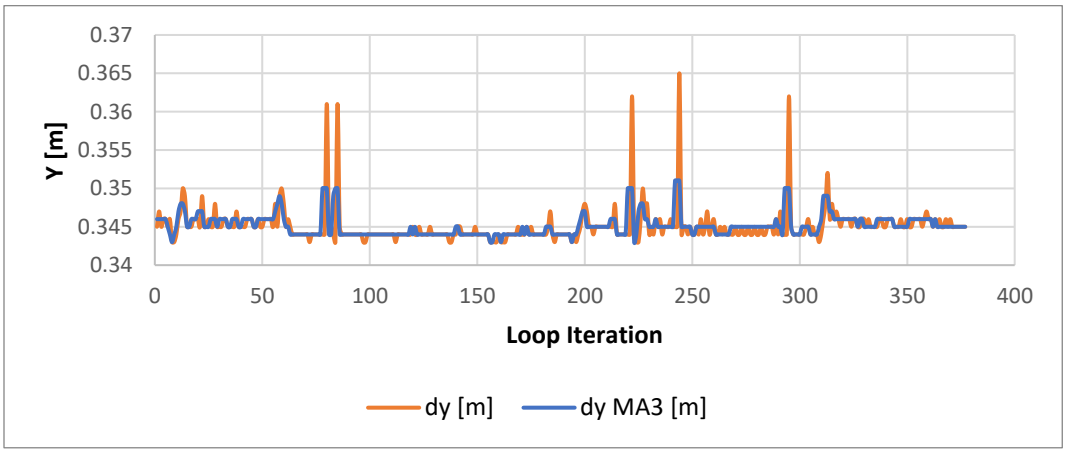


Figure 3.24 Raw data and 3-point moving average filter of Y position data in the camera frame.

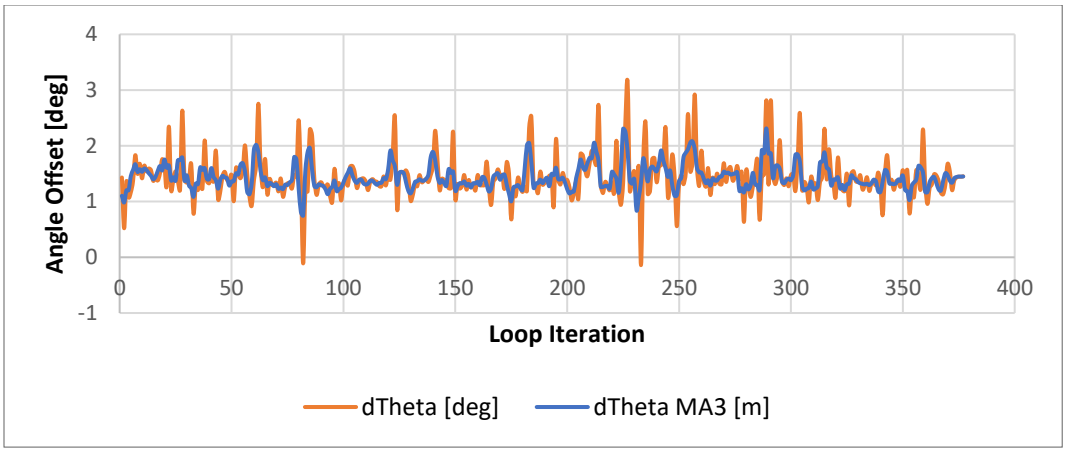


Figure 3.25 Raw data and 3-point moving average filter of orientation data in the camera frame.

3.3 Simulator Software

All the code below is computed based on plane transformations as the motion of simulators and manipulator is limited to the surface of granite table. All the offsets and transformations can easily be expanded into 3D space.

3.3.1 Servo Motor Driver Software

To control the servomotors, the Arduino, LabView library interface was used to communicate between the onboard computer and the Arduino. The library has a built-in driver to send the joint angles and convert them to the PWM signal. The driver is limited to 1-degree increments and does not take advantage of the 0.3-degree resolution of the servo motors. To adjust for this the PWM driver for the LabView Interface was used. Unfortunately, the datasheets for the servo motors did not provide the PWM signal to actuate the motors. The motors did not operate correctly with the conventional PWM range

used for servo motors. To determine the PWM range for the servo motors the driver that accepts degrees for input was swept for the operational range of the motor. Then the driver that reads the PWM signal to the motor was used to see what the LabView driver, operated in degrees, outputs to the servo motors. The resultant range for the PWM signal was 544ns corresponding to 0 degrees and 1856ns corresponding to 180 degrees. This was then used to get a finer resolution to the motors using the PWM driver.

3.3.2 Robotic Manipulator Path Planning Algorithm

To make the manipulator move to a target position and orientation, one must compute the required joint angles of the manipulator. To obtain the joint angles the inverse kinematics of the manipulator must be computed. The manipulator is simple without redundant degrees of freedom so that the inverse problem can be solved simply with the geometry of the manipulator. Here, as the position and attitude of the base simulator is controlled, the manipulator is treated as having a “fixed base” relative to the chaser simulator. With this, the simulator and the manipulator are independently controlled, and any moments or torques the manipulator transmits to the chaser simulator are treated as external disturbances that the chaser simulators controller will then correct. To compute the inverse kinematics of the manipulator the desired position and orientation of the manipulator must be known. The parameters of the manipulator must also be known and are listed below.

First, the current open-loop position and orientation of the end-effector is computed in the robotic manipulator reference frame, see Figure 3.26.

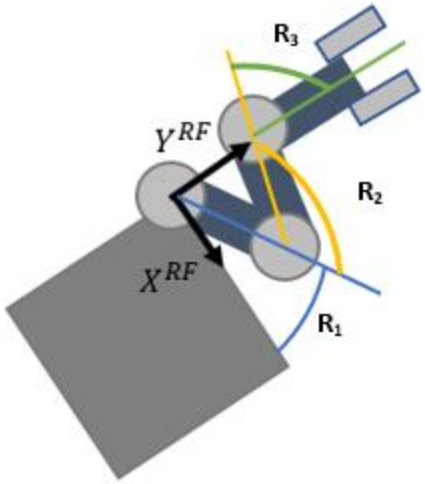


Figure 3.26 Reference from of the robotic manipulator.

$$X_R^{RF} = L_1 \cos(R_1) + L_2 \cos(R_1 + R_2) + L_3 \cos(R_1 + R_2 + R_3) \tag{3.5}$$

$$Y_R^{RF} = L_1 \sin(R_1) + L_2 \sin(R_1 + R_2) + L_3 \cos(R_1 + R_2 + R_3) \tag{3.6}$$

$$\theta_R^{RF} = R_1 + R_2 + R_3 \tag{3.7}$$

Then the target's position is computed in the robotic reference frame. To do this, the position and orientation of the Aruco marker are required. Then, a transformation from the Aruco marker to the target's capture point is performed. Using the known relative position and orientation of the marker and the target's capture point, this transformation can be computed. These transformations are then used to compute the difference between the target simulator's capture point and the end-effector of the robotic manipulator.

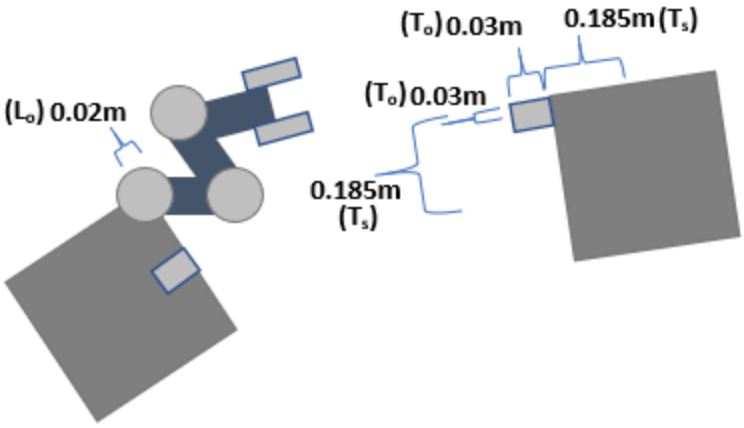


Figure 3.27 Parameters required for frame conversions.

The target capture point is computed in the robotic manipulator frame as:

$$X_{CP-EE}^{RF} = X_c^{CF} - X_R^{RF} - T_s \cos(\theta_{TS}^{WF}) - T_o \sin(\theta_{TS}^{WF}) + T_s \tag{3.8}$$

$$Y_{CP-EE}^{RF} = Y_c^{CF} - Y_R^{RF} + T_s \sin(\theta_{TS}^{WF}) - T_o \sin(\theta_{TS}^{WF}) + L_0 \tag{3.9}$$

where T_s is equal to half the side length of target simulator (0.185m) and the capture fixture with an offset of 0.03m, T_o, is in both the X and Y direction of the target simulator representing the offset of the target capture fixture. L_0, is the additional offset of 0.02m denoted for the difference in the attachment point between the tracking camera and the robotic manipulator in the body fixed from of the chaser simulator.

Next, to track the distance to the target, the manipulator will move in a step of 0.5 mm, d_s, towards the target. The magnitude of the distance between the target and the end-effector is used to determine the x and y increments of the end-effector in the next step from the current position.

$$d_\tau = \sqrt{(X_{CP-EE}^{RF})^2 + (Y_{CP-EE}^{RF})^2} \tag{3.10}$$

$$d_x = d_s \frac{X_{CP-EE}^{RF}}{d_\tau} \tag{3.11}$$

$$d_y = d_s \frac{Y_{CP-EE}^{RF}}{d_\tau} \tag{3.12}$$

The next position for the manipulator is computed by adding the required increments in the x and y directions to the current position of the manipulator end-effector in the robotic frame. This causes the manipulator to traverse the closest straight-line path to the target at the required orientation.

$$X_R^{RF} = X_R^{RF} + d_x \tag{3.13}$$

$$Y_R^{RF} = Y_R^{RF} + d_y \tag{3.14}$$

Using the next position for the manipulator end-effector, the next position for the R_3 joint must be computed. This uses the desired orientation of the R_3 joint to be 90^o and adding the current orientation of the target.

$$X_{R3}^{RF} = X_R^{RF} - L_3 \cos(\frac{\pi}{2} + \theta_c^{CF}) \tag{3.15}$$

$$Y_{R3}^{RF} = Y_R^{RF} - L_3 \sin(\frac{\pi}{2} + \theta_c^{RF}) \tag{3.16}$$

Using the desired position for the R_3 joint, the inverse kinematics of the first two links can be computed using geometry. Using the law of cosines, the second joint angle can be computed.

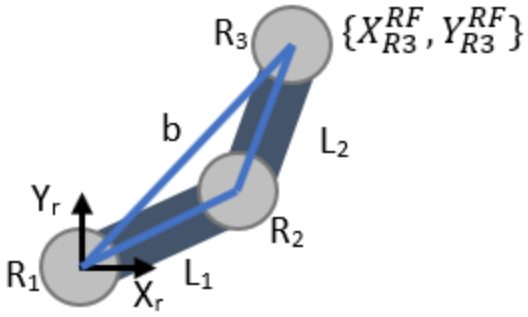


Figure 3.28 Geometry used to determine the angle of the R_2 joint of the robotic manipulator.

$$b^2 = L_1^2 + L_2^2 - 2L_1L_2\cos(\pi - R_2) = (X_{R3})^2 + (Y_{R3})^2 \tag{3.17}$$

$$-\cos(\pi - R_2) = \frac{(X_{R3})^2 + (Y_{R3})^2 - L_1^2 - L_2^2}{2L_1L_2} \tag{3.18}$$

$$c\,\text{os}(R_2) = \frac{(X_{R3})^2 + (Y_{R3})^2 - L_1^2 - L_2^2}{2L_1L_2} = c_2 \tag{3.19}$$

There are two potential solutions for R_2. Using trigonometric identities, it is possible to obtain a solution for both permutations of R_2. Due to the limited motion of the second servo motor, only one of the two permutations are physically possible. This avoids the problem where the system can have two solutions and avoids singularities when crossing the point where the manipulator links are all aligned.

$$\sin^2(R_2) + \cos^2(R_2) = 1 \tag{3.20}$$

$$\tan(R_2) = \frac{\sin(R_2)}{\cos(R_2)} \tag{3.21}$$

Considering the specific configuration of the manipulator used in this experimentation, only the negative sign of square root is used.

$$R_2 = \tan^{-1}\left(\frac{-\sqrt{1-c_2}}{c_2}\right) \tag{3.22}$$

To check if the solution exists, the following condition must be met. If the condition fails, then the manipulator will remain in its current position and maintain the latest joint positions.

$$1 - c^2 \geq 0 \tag{3.23}$$

Now using R_2 it is possible to compute the value of R_1 by a set of right-angle triangles, see Figure 3.29.

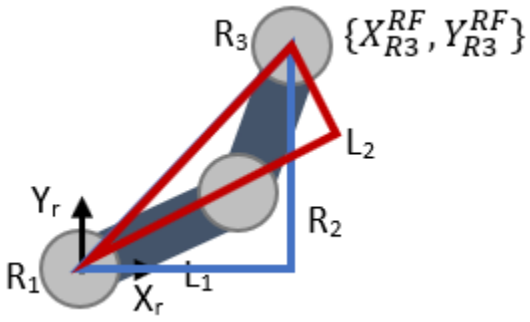


Figure 3.29 Geometry to solve inverse kinematics of first two joints.

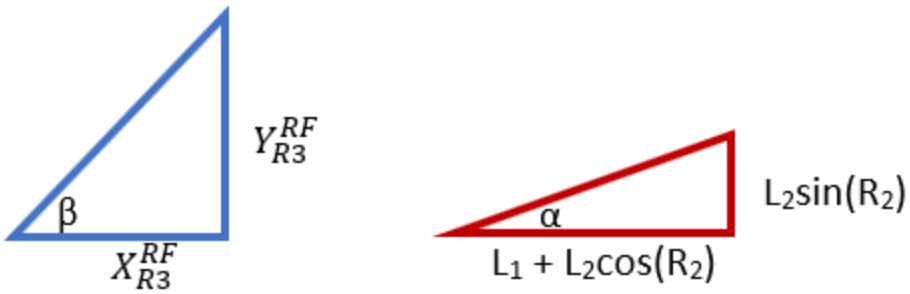


Figure 3.30 Triangles obtained from geometry of first two joints.

$$\beta = \tan^{-1}\left(\frac{Y_{R3}^{RF}}{X_{R3}^{RF}}\right) \tag{3.24}$$

$$\alpha = \tan^{-1}\frac{L_2\sin(R_2)}{L_1 + L_2\sin(R_2)} \tag{3.25}$$

The first joint angle is then computed as the difference between the angles of the two right angle triangles.

$$R_1 = \beta - \alpha \tag{3.26}$$

The third joint angle is then computed as the desired orientation subtracting the previous two joint angles.

$$R_3 = 90 - \theta_c^{RF} - R_1 - R_2 \tag{3.27}$$

3.3.3 Gripper Activation

There are two ways to initiate the grasping of the target fixture. The first is through the wire contact sensor, where a capture signal is recorded. If this signal is received, the gripper will immediately close to capture the target.

The second method is if the gripper is within 0.0125m of the target computed through forward kinematics then the gripper is to close slowly at a rate of 10 deg/s. The value of 0.0125m is the waypoint switching distance and the chaser simulator may have difficulty converging on the final waypoint in the system.

3.3.4 Simulator Path Planning Algorithm

To determine how the simulator will approach and synchronize with the target, a path planning procedure must be determined. To keep the path computationally simple, the closest straight-line path is used to make the far-range approach to the target. Here, the far-range approach is any distance greater than the 0.75m between the target and chaser simulators. Then a circular synchronization manoeuvre is used for the face with the robotic manipulator to align with the capture face of the target simulator. To traverse, waypoints are spaced apart 1mm along the path with a waypoint switching distance of 1.25cm. This waypoint spacing was chosen so that the chaser simulator will follow the desired path closely. The switching distance was chosen to be larger than the measurement error of the star tracker positioning system using the raw data measurements.

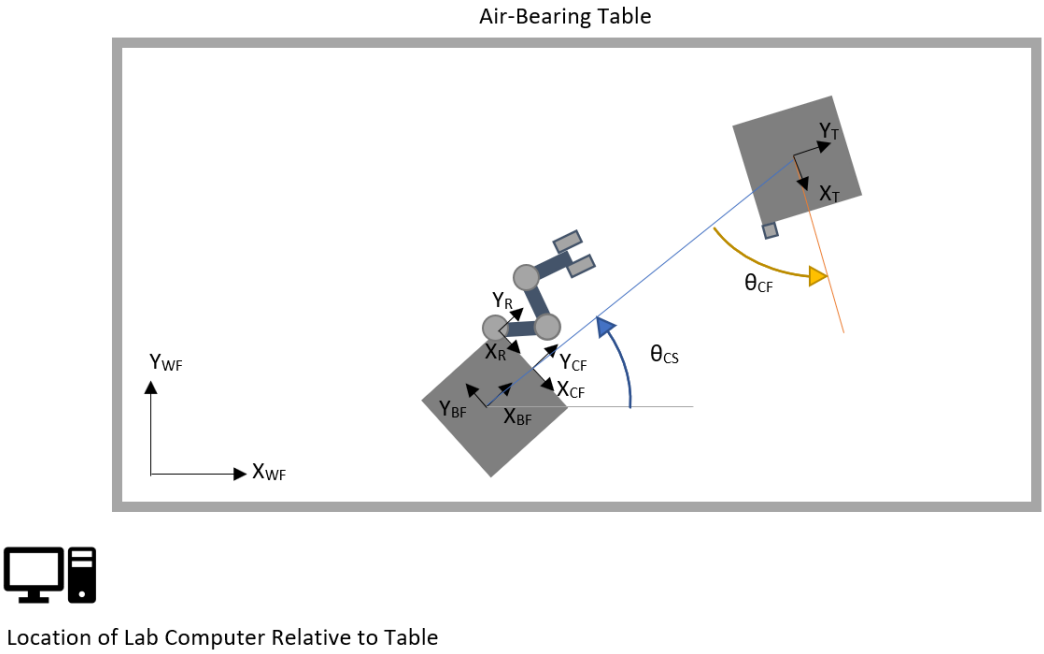


Figure 3.31 Table representation.

To compute the next waypoint along the straight-line path, 1mm is added towards the target from the latest waypoint to create the new target waypoint. The difference between the chaser simulator's (CS) current position and the target simulator (TS) position (see Figure 3.32) is computed as:

$$dx^{WF} = X_{CS}^{WF} - X_{TS}^{WF} \tag{3.28}$$

$$dy^{WF} = Y_{CS}^{WF} - Y_{TS}^{WF} \tag{3.29}$$

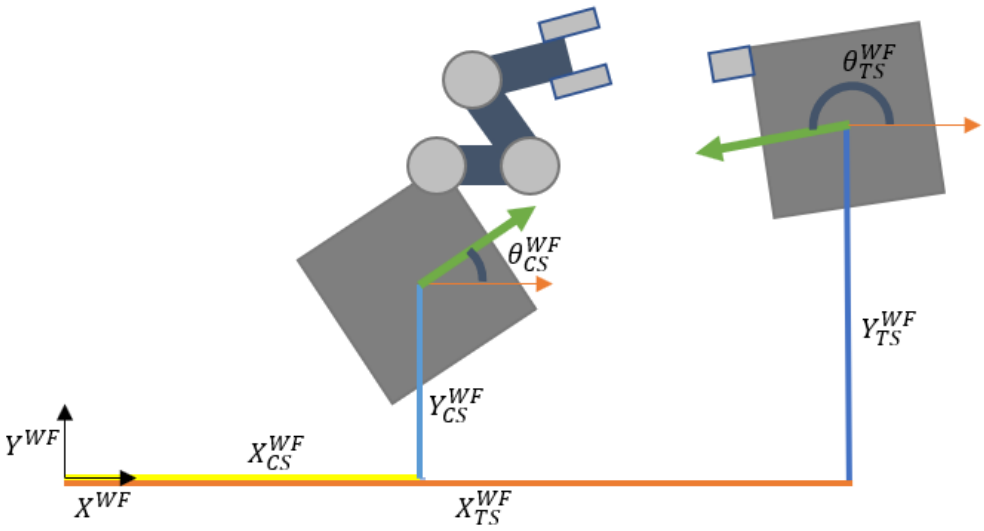


Figure 3.32 Position and orientation of simulators in world frame.

The orientation between the chaser satellite to the target in the world frame (see Figure 3.33) is:

$$\varphi^{WF} = \arctan 2(dy^{WF}, dx^{WF}) \tag{3.30}$$

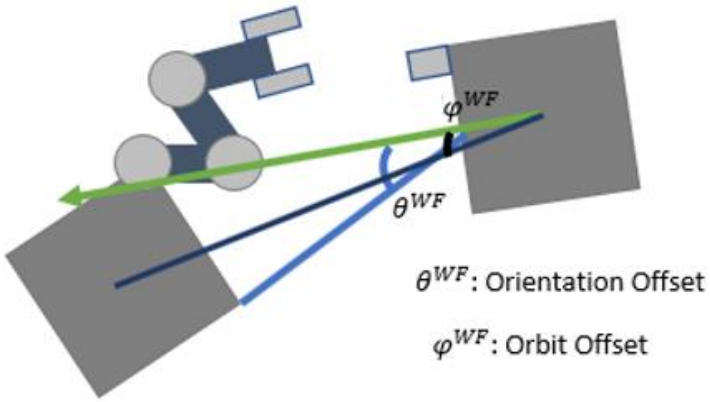


Figure 3.33 Chaser simulator's orbit and orientation offsets.

The next position waypoint for the chaser satellite is computed as:

$$X^{WF} = x^{WF} - 0.001\cos(\varphi^{WF}) \tag{3.31}$$

$$Y^{WF} = y^{WF} - 0.001\sin(\varphi^{WF}) \tag{3.32}$$

where x^{WF} and y^{WF} are the current waypoint for the chaser simulator and X^{WF} and Y^{WF} are the next waypoint position.

The orientation waypoints are positioned 0.074 deg apart. The difference between the current orientation waypoint and the next waypoint is computed as:

$$dt = d\varphi^{WF} + \pi + zr^{WF} \tag{3.33}$$

$$Zr^{WF} = zr^{WF} - 0.074 * sign(dt) \tag{3.34}$$

where zr^{WF} is the current orientation waypoint and Zr^{WF} is the next waypoint.

When both the position of the simulator reaches the synchronization path and the orientation of the chaser simulator is lined up with the target simulator, the chaser will begin the synchronization phase. Once in the synchronization phase, the chaser simulator will follow a circular path with a radius of 0.75m with the target simulator at its centre. This radius was chosen as the closest distance the chaser simulator can orbit the target without the manipulator interfering with the target simulator when in its ready position.

The orientation waypoints are spaced out 0.074 deg apart. 0.074 deg was chosen so that along the circular path with a radius of 0.75m the rotation of 0.074 deg would place the waypoints 1mm apart. The waypoint switching distance was chosen as 1.25 deg which is larger than the measured angular noise.

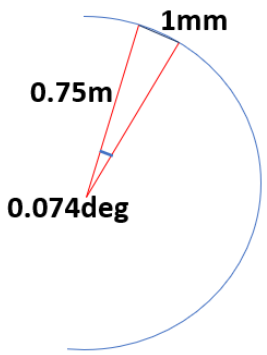


Figure 3.34 Computation of waypoints for synchronization orbit.

When the target is spinning, the waypoint is projected further down the path by three loop iterations of the target's angular velocity. Without performing this prediction step, the chaser will always lag the target and would never capture.

$$X^{WF} = 0.75\cos(d\varphi^{WF} - 0.3T_{zw} - 0.074sign(d\varphi^{WF})) + X_{TS}^{WF} \tag{3.35}$$

$$Y^{WF} = 0.75\sin(d\varphi^{WF} - 0.3T_{zw} - 0.074sign(d\varphi^{WF})) + Y_{TS}^{WF} \tag{3.36}$$

$$Zr^{WF} = zr - 0.3T_{zw} - 0.074sign(d\varphi^{WF}) \tag{3.37}$$

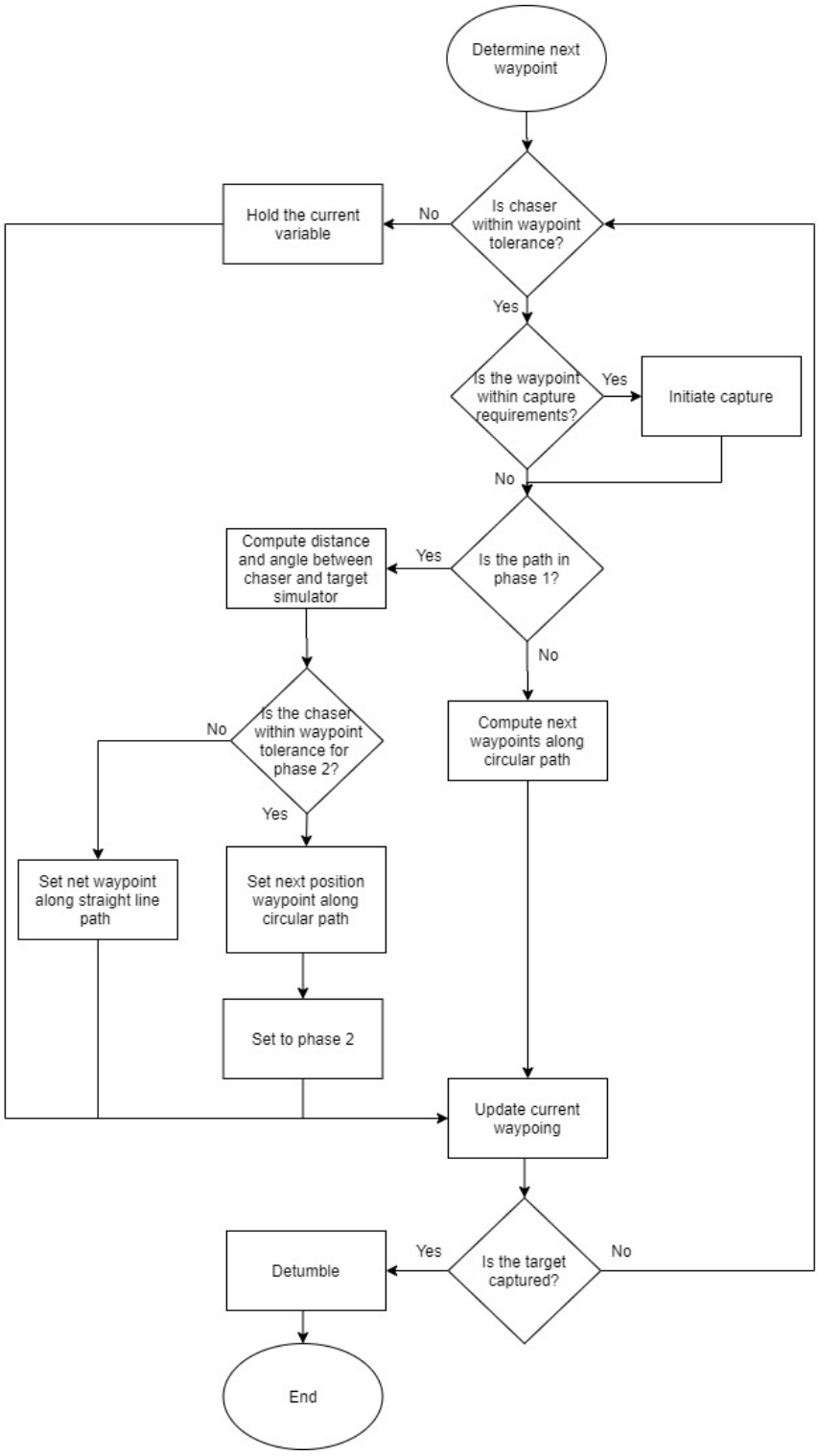


Figure 3.35 Path planning algorithm sequence.

For the robotic manipulator to initiate the capture process, the chaser simulator must satisfy two conditions. The first is the offset orientation of the two simulators (chaser and target) must be less than ±1.25 degrees, see Figure 3.34. This corresponds to the acceptable orientation waypoint error. The orientation offset could be increased to ±6 degrees before the limitation of the servomotor's accuracy becomes a significant issue. This is limited here to ensure that the inaccuracy of the servo motors will not cause the path of manipulator to deviate from the target.

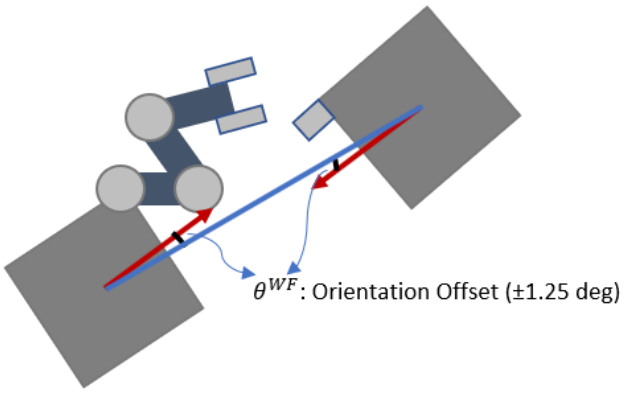


Figure 3.36 Orientation offset of the two simulators.

The second condition for capture is the position of the chaser simulator around the target simulator while performing the synchronization path, see Figure 3.37. This was chosen to be ±3.5 degrees, as at a radius of 0.75m, this would result in an offset of ±3.75cm, 3 times the acceptable tolerance for positioning of the manipulator. This value was chosen, as a 3-point moving average is used and could cause the chaser to fall behind by 3 loop iterations which corresponds to 3 times the acceptable tolerance. This

synchronization offset can be increased to ±5 degrees before the servomotor's accuracy becomes a significant issue.

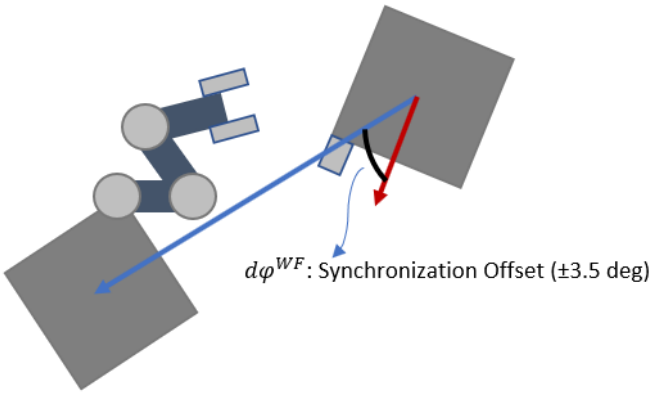


Figure 3.37 Synchronization Offset of the two simulators.

3.3.5 Detumbling Algorithm

After the tumbling target simulator has been successfully captured, both the target and chaser are tumbling together. The next task of the chaser simulator is to detumble the target/chaser pair. The challenge with this is that if a single waypoint is selected, and the target and the chaser simulators both move far from this point, the pair will attempt to travel back to this point, causing significant overshoot. The reason for this is the simulator is trying to hold that single waypoint regardless of how far it moves away from it. At the point the simulator pair reaches an angular velocity of zero, it may have moved well beyond that desired waypoint. As that waypoint is still the desired position after stabilization, the pair will attempt to move back to that desired point.

To remedy this, if the chaser exceeds the waypoint tolerance discussed in the path planning algorithm (1.25cm and 1.25 degrees), then a new waypoint is selected based on the chaser simulator's current position. This allows the chaser to detumble the target without needing to converge to the original capture waypoint.

3.3.6 Thruster Allocation

The base spacecraft simulator can keep itself stable both in position and orientation. Thus, the manipulator can be treated as on a fixed base to simplify the inverse kinematics of the manipulator. Any transfer of momentum from the manipulator to the base simulator is treated as a disturbance.

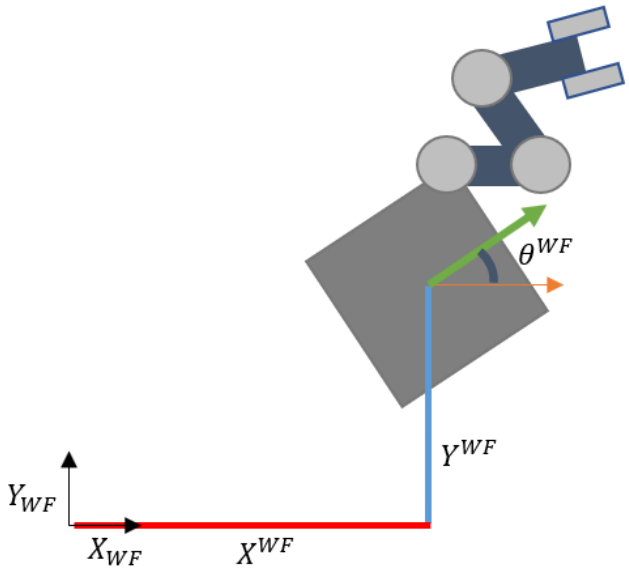


Figure 3.38 World frame coordinates of the chaser simulator.

The components that make the position and orientation are represented as a vector containing, the position and orientation components as shown in Figure 3.38.

$$\mathbf{q} = \begin{bmatrix} X^{WF} \\ Y^{WF} \\ \theta^{WF} \end{bmatrix} \tag{3.38}$$

Due to the fact that the simulator uses eight on-off unidirectional thrusters with a constant thrust, see Figure 3.39, the simulator must map the required forces and torques to the eight thrusters. The forces are mapped to the thrusters using the mapping matrix **C.** Each row maps the thrusters to the x, y translational and rotational components respectfully. **R** is the vector representing the forces produced by the eight on-off unidirectional thrusters [80,90].

$$\mathbf{F} = \mathbf{CR} \tag{3.39}$$

$$\mathbf{C} = \begin{bmatrix} -c3 & s3 & s3 & c3 & c3 & -s3 & -s3 & -c3 \\ -s3 & -c3 & -c3 & s3 & s3 & s3 & c3 & -s3 \\ r_j & -r_j & r_j & -r_j & r_j & -r_j & r_j & -r_j \end{bmatrix} \tag{3.40}$$

$$\mathbf{R} = \begin{bmatrix} R_1 & R_2 & R_3 & R_4 & R_5 & R_6 & R_7 & R_8 \end{bmatrix}^T \tag{3.41}$$

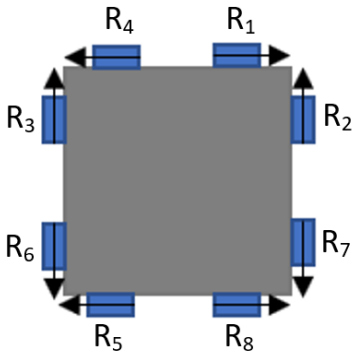


Figure 3.39 Thruster mapping.

This can be simplified by grouping thrusters that would produce opposing force and torque values together [90]. This creates four on-off bi-directional thrusters that can produce positive and negative force values. The thrusters are then mapped back to the original unidirectional thrusters using the sign of the required thrusting force. If the required thrust is positive the first of the pair would activate. If the required thrust is negative the second thruster of the pair would activate. This simplifies the above equations requiring a smaller inverse calculation and adding of a simple thruster mapping.

$$\mathbf{C} = \begin{bmatrix} -c3 & s3 & c3 & -s3 \\ -s3 & -c3 & s3 & c3 \\ r_j & r_j & r_j & r_j \end{bmatrix} \tag{3.42}$$

$$\mathbf{R} = \begin{bmatrix} R_1 & R_2 & R_3 & R_4 \end{bmatrix}^T \tag{3.43}$$

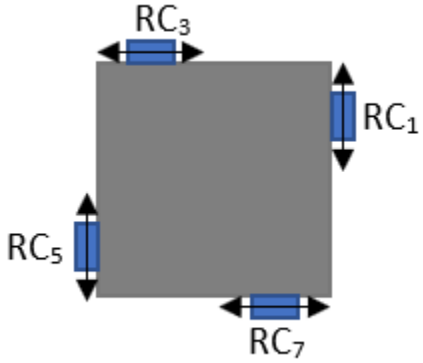


Figure 3.40 Simplified thruster mapping.

Taking the pseudo-inverse of the mapping matrix **C,** and multiplying it by the required generated forces F, the required forces produced by the thrusters are computed. The pseudo-inverse is denoted by † [90].

$$\mathbf{R} = \mathbf{C}^{\dagger}\mathbf{F} \tag{3.44}$$

However, as the thrusters are on-off with a single force value, the required force must be thresholded or discretized into binary states: zero and constant force. The thrusters are calibrated to produce a force of R_{th}=0.065N. Therefore, if the required force for the thruster equals or exceeds the threshold then the thruster is turned on. If it is less than the threshold the thruster remains off.

Table 3.3 Thruster mapping for the chaser simulator

Thruster Mapping	**Output**
$\boldsymbol{RC_1 \geq R_{th}}$	R_7
$\boldsymbol{RC_1 \leq -R_{th}}$	R_8
$\boldsymbol{RC_2 \geq R_{th}}$	R_1
$\boldsymbol{RC_2 \leq -R_{th}}$	R_2
$\boldsymbol{RC_3 \geq R_{th}}$	R_3
$\boldsymbol{RC_3 \leq -R_{th}}$	R_4
$\boldsymbol{RC_4 \geq R_{th}}$	R_5
$\boldsymbol{RC_4 \leq -R_{th}}$	R_6

The thrusters are activated by writing to a National Instruments NI USB-6212 Data Acquisition Card. Through the LabVIEW driver a 16-bit string must be written to the data acquisition card where each bit in the string signifies one of the digital pins being activated. As all the bits must be written to at the same time the first 7 bits are zero, followed by the float activation, and then by 8 bits for each thruster.

Address Number	Address Representation	ON (1)/OFF (0)
0	Unwired Placeholder	0
1	Unwired Placeholder	0
2	Unwired Placeholder	0
3	Unwired Placeholder	0
4	Unwired Placeholder	0
5	Unwired Placeholder	0
6	Unwired Placeholder	0
7	Air-Bearing Float	1/0
8	Thruster 1	1/0
9	Thruster 2	1/0
10	Thruster 3	1/0
11	Thruster 4	1/0
12	Thruster 5	1/0
13	Thruster 6	1/0
14	Thruster 7	1/0
15	Thruster 8	1/0

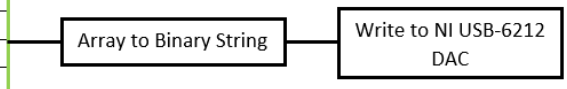


Figure 3.41 Writing 16-bit string to NI USB-6212 DAC.

Chapter 4 CONTROL STRATEGIES

Summary: This chapter contains three sections. The first is a discussion of the two control strategies chosen and control strategies commonly used on these testbeds. The second section is the PD control strategy. It includes the controller design and stability requirements for the PD controller. The section also includes the controller stabilities designed for the target simulator. The third section is the controller design for the adaptive controller using fuzzy gain scheduling to enhance the PD controller.

4.1 Control Strategies

For this paper two control strategies were chosen; A PD controller, and an adaptive controller. The two controllers were chosen as they were not complex and would sufficiently demonstrate how different control algorithms can be demonstrated on the platform.

Commonly implemented controllers for OOS on air-bearing tables are PD control [76,80,90] and impedance control. Impedance control is commonly used in the control of robotic manipulators. There are various impedance control methods implemented [80,91–94]. These implementations require feedback to determine the end-effectors position and

orientation in the inertial frame [80,91]. The manipulator implemented in this testbed does not have any feedback for the manipulator and this control scheme could not be implemented. The PD controller was simple to implement and allowed the feed-forward of the robotic manipulator.

The adaptive controller was chosen here as the minimum implementation to improve the PD controller on the testbed. As the simulator uses on-off thrusters poses the challenge that the same set of thrusters are used for both the translational and rotational components of the system. This poses a problem as the PD controller implementation has an underactuated region when one of its parameters is close to a waypoint. When a thruster is activated to correct the translational component, it will apply a torque onto the system causing it to rotate. The simulator constantly oscillates between correcting translational and rotational motions. The adaptive controller implemented here allows for the thruster activation in the previously underactuated region of the PD controller, allowing the controller to hold desired waypoints more closely supressing underactuated oscillations.

4.2 PD Control Strategy

The position and orientation of the simulator are represented by a state vector $\boldsymbol{q}$, such that.

$$\mathbf{q} = \begin{bmatrix} X^{WF} \\ Y^{WF} \\ \theta^{WF} \end{bmatrix} \tag{4.1}$$

where X^{WF}, Y^{WF}, and θ^{WF} are the position and orientation of the simulator in the world frame.

To control the simulator's state, a PD controller was implemented [80,90]. A PD controller uses the difference in the desired, $\mathbf{q_d}$ and the current position, $\mathbf{q}$ of the simulator as well as the difference in their velocities to determine the required forces and torques for the simulator. The differences between the desired and current values produce a corresponding error.

$$\mathbf{e} = \mathbf{q}_d - \mathbf{q} \quad \text{and} \quad \dot{\mathbf{e}} = \dot{\mathbf{q}}_d - \dot{\mathbf{q}} \tag{4.2}$$

The required control thrust $\boldsymbol{F}$ is determined by a PD controller, such that,

$$\mathbf{F} = \mathbf{M}(\mathbf{K}_p\mathbf{e}(t) + \mathbf{K}_v\dot{\mathbf{e}}(t)) \tag{4.3}$$

$$\mathbf{M} = \begin{bmatrix} M \\ M \\ I \end{bmatrix} \tag{4.4}$$

$$\mathbf{K}_p = \begin{bmatrix} K_{px} \\ K_{py} \\ K_{p\theta} \end{bmatrix} \tag{4.5}$$

$$\mathbf{K}_v = \begin{bmatrix} K_{vx} \\ K_{vy} \\ K_{v\theta} \end{bmatrix} \tag{4.6}$$

$$\mathbf{F} = \begin{bmatrix} F_x \\ F_y \\ \tau \end{bmatrix} \tag{4.7}$$

where K_p and K_v are the control gains for position and velocity, F and τ are the control force and torques, $\mathbf{M}$ is a matrix of the mass of the simulator M and its moment of inertia I.

The new state of the simulator is determined by using Newton's second law. This is done by obtaining the resultant acceleration of the system and then integrating it twice to obtain the simulators new position, see Figure 4.1.

$$\ddot{\mathbf{q}} = \frac{\mathbf{F}}{\mathbf{M}} \tag{4.8}$$

$$\mathbf{q} = \iint \ddot{\mathbf{q}} \tag{4.9}$$

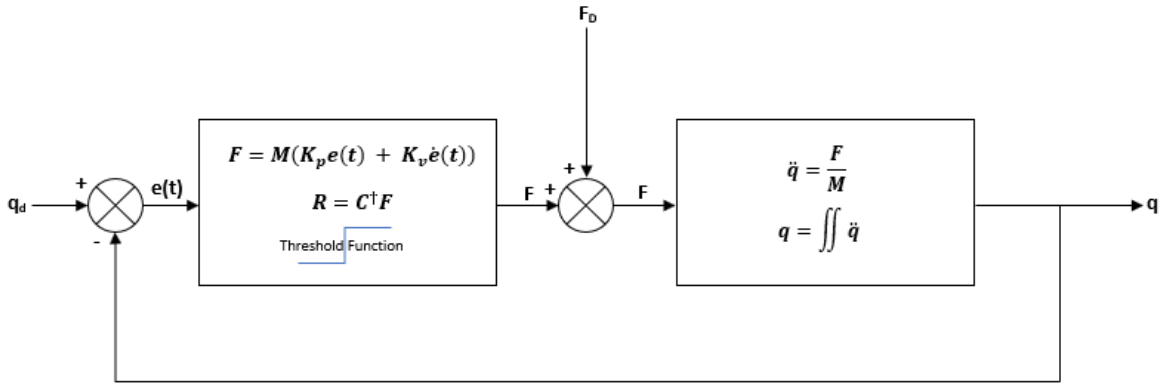


Figure 4.1 Control loop for the chaser simulator.

It is important to note that this controller design is not limited to the control of the base simulator. If feedback was available on the robotic manipulator joints, then the controller can be expanded to include the joints and a single path planning solution can be used to control the manipulator-base simulator system.

4.2.1 Stability of PD Controller

Next, we examine the stability of the PD controller to determine the stable region of the gains K_p and K_v. First, let us convert our control loop into the Laplace domain to obtain the transfer function as shown in Figure 4.2.

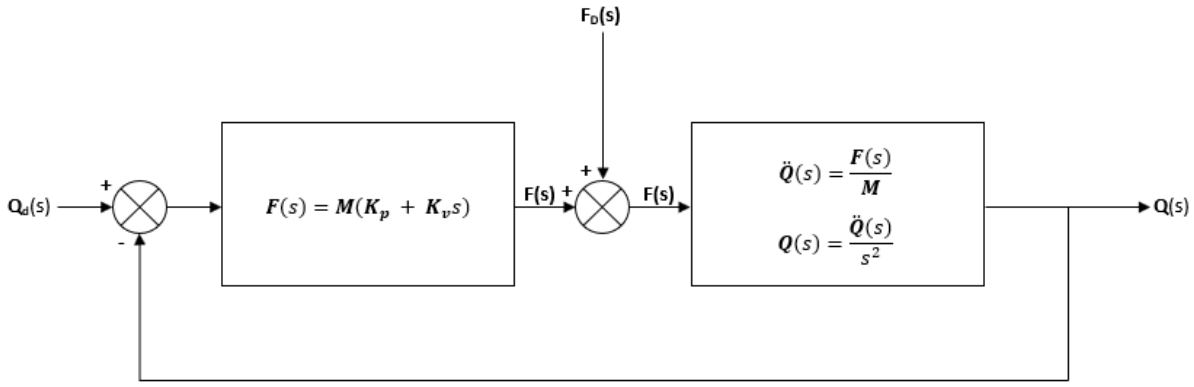


Figure 4.2 PD chaser controller using Laplace domain.

$$T(s) = \frac{K_v s + K_p}{s^2 + K_v s + K_p} \tag{4.10}$$

To assess the stability of the system, we only need to find the roots of the denominator of the transfer function, such that,

$$D(s) = s^2 + K_d s + K_p = 0 \tag{4.11}$$

Using the Routh-Hurwitz criterion, the stability of the system can be determined. The coefficient n^{th} order term is placed in the left-hand column and the $n\text{-}2^{th}$ term is placed next to it. The final n^{th} term here is computed as the negative determinate of the above four terms divided by the term directly above it. By assessing the sign changes of the n^{th} term,

we can determine whether the poles of the transfer function are in the left or right-half plane. If any of the poles are in the right-half plane, the system is unstable. A pole is denoted in the right-hand plane if there is a sign change of the n^{th} term. As we can see if either K_v or K_P are negative, a pole will be in the right half plane and the system will become unstable. Therefore, the gains must both be positive to be stable.

$$\frac{-\begin{vmatrix} 1 & K_p \\ K_v & 0 \end{vmatrix}}{K_v} = K_p \tag{4.12}$$

Table 4.1 Routh table to determine the stability of the PD controller

Sign Change	Order of Term	n	n-2
	s^2	1	K_p
+	s^1	K_v	0
+	s^0	K_p	

The optimal values to choose for the gains can be further assessed by examining the pole placement of the transfer function. The system will perform optimally when it is critically dampened. When the system is critically damped there will be no overshoot and the rise time will be maximized. A critically damped system with a second order denominator in its transfer function will have two overlapping poles. To determine this, we plug the coefficients of the transfer function denominator into the quadratic formula.

$$s = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-K_v \pm \sqrt{K_v^2 - 4K_p}}{2} \tag{4.13}$$

For the quadratic formula to have a double root, the square-root term must become zero. We then find a ratio between the K_p and K_v gains for the system to become critically damped.

$$K_v^2 - 4K_p = 0 \tag{4.14}$$

$$K_v = 2\sqrt{K_p} \tag{4.15}$$

Therefore, for the chaser simulator we have two requirements for the system to be stable:

1) The gains K_v and K_p must both be positive.

$$K_v > 0 \quad K_p > 0 \tag{4.16}$$

2) The gains K_v and K_p must have the following ratio to be critically damped.

$$K_v = 2\sqrt{K_p} \tag{4.17}$$

Next, we determine specific values for K_v and K_p for our system. As the system is critically dampened, ζ=1. Looking at the general second-order system we can determine the frequency component of the settling time. Comparing the denominator of the general second-order system and the transfer function for the robotic simulator, the frequency component is determined to be $\omega = \sqrt{K_p}$.

$$\textit{General}\ \text{Second-Order System} = \frac{\omega^2}{s^2 + 2\zeta\omega s + \omega^2} \tag{4.18}$$

Looking at the time response of the system, the larger the K_v and K_p gains the smaller the settling time of the system. Values of K_v and K_p at infinity would result in the smallest settling time.

$$T_s = \frac{4}{\zeta\omega} = \frac{4}{\sqrt{K_p}} \tag{4.19}$$

$$\lim_{K_p \to \infty} T_s = \lim_{K_p \to \infty} \frac{4}{\zeta\omega} = \lim_{K_p \to \infty} \frac{4}{\sqrt{K_p}} = 0 \tag{4.20}$$

However, due to the nature of the system using on-off thrusters, the large values for the gains would result in the thrusters always being on. Therefore, we must choose values for the gains carefully. To determine what the gains we consider two terms, the waypoint switching distance and the force output of the thrusters. The waypoint switching distance was specifically chosen to be larger than the error of the star tracking system. The force output of the thrusters is 0.065 N each. Therefore, the gain of the system should be chosen so that if the simulator is at or beyond the waypoint switching distance, the thruster will be activated.

To compute these gains for the translational controller, we compare the K_p for the waypoint switching distance and the thruster force. We divide the output force of the thruster by the mass of the simulator and the waypoint switching distance. We obtain a value of K_p of 0.26. Using the required ratio to ensure the system is critically damped, the gain K_v can be determined. The K_v gain is computed as 1.02. The resultant poles for the system can be seen in Figure 4.3.

$$F_t = MK_p d_s \tag{4.21}$$

$$0.065N = 20kg * K_p * 0.0125m$$

$$K_p = 0.26$$

$$K_v = 2\sqrt{K_p} = 2\sqrt{0.26} = 1.02$$

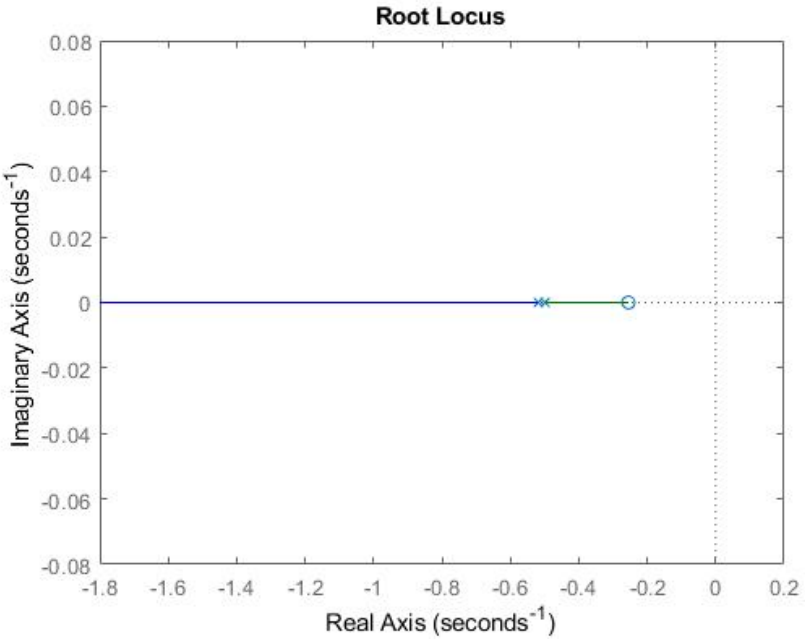


Figure 4.3 Root-Locus for the PD position controller.

The gains for the attitude controller must also be computed. We must first compute the torque the thruster will induce on the system. The thrusters are placed 1/3rd the side length from the edge of the simulator. The radius from the center of mass of the simulator is 0.39m at an angle of 71.6^{o}, see Figure 4.4. Using the torque of the thrusters and the inertia of the system the gains are K_p =2.41 and K_v = 3.11. The resultant poles for the system can be seen in Figure 4.5.

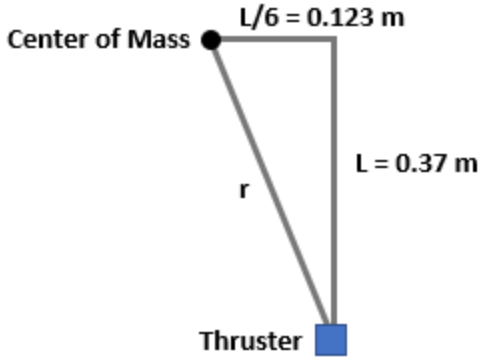


Figure 4.4 Location of thruster with respect to the center of the robotic simulator.

$$\tau = Fxr = 0.065N * (\sqrt{0.123^2 + 0.37^2})m * \sin\left(\arctan\left(\frac{0.37}{2}\frac{6}{0.37}\right)\right) = 0.024Nm$$

(4.22)

$$\tau = IK_p e(t) \tag{4.23}$$

$$I = \frac{Ms^2}{6} = \frac{20kg * 0.37m^2}{6} = 0.456kgm^2$$

$$0.024Nm = 0.456kg\ m^2\ \mathrm{K}_p\left(\frac{1.25\deg}{180\deg}\pi\right)$$

$$K_p = 2.41$$

$$K_v = 2\sqrt{K_p} = 2\sqrt{2.41} = 3.11$$

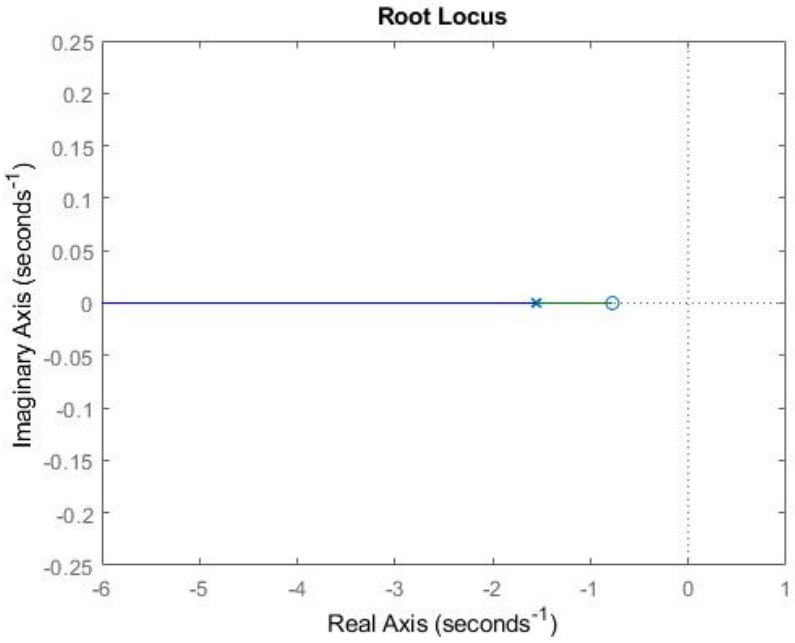


Figure 4.5 Root Locus for PD angular position controller.

4.2.2 Target Simulator Stability

The target simulator has a different controller implemented to simulate a tumbling target. The target is equipped with a reaction wheel that is used to rotate the target simulator with a constant velocity. This is chosen over the chaser's thruster PD controller as it can continuously adjust the target's rotation whereas the thrusters are on-off and cannot make small and smooth adjustments in the angular velocity.

The reaction wheel was not used on the chaser simulator as the motion of the robotic manipulator caused the reaction wheel to over-spin. This is primarily due to the reaction wheel installed on the simulators being too small for the inertia of the robotic simulator.

When the robotic manipulator makes large motions, or experiences large disturbances it caused the reaction wheel to be saturated. This occurred when the test case began where the manipulator would reset its motion and when the manipulator actuates towards and grasps the target. The reaction wheel could have been programmed to activate only after a delay, after the manipulator resets and turned off prior to the manipulator use for capture. The reaction wheel could provide more accurate pointing than the thrusters during the manoeuvre phase however, this level of accuracy is not necessary at this phase. This level of accuracy is most crucial when the manipulator is actuated attempting capture the target, however, became saturated. In the future, a more massive reaction wheel can be installed that can overcome the disturbances caused by the manipulator and be used during the capture phase.

The target simulator has the PD thruster controller and a separate P reaction wheel controller to maintain constant angular velocity. The PD controller is like that of the chaser simulator without rotational component.

$$\mathbf{C} = \begin{bmatrix} -c3 & s3 & c3 & -s3 \\ -s3 & -c3 & -c3 & c3 \end{bmatrix} \tag{4.24}$$

$$\mathbf{R} = [R_1 \quad R_2 \quad R_3 \quad R_4]^T \tag{4.25}$$

Taking the pseudo-inverse of the mapping matrix **C,** and multiplying it by the required generated forces **F**, the required forces produced by the thrusters are computed. The pseudo-inverse is denoted by †.

$$\mathbf{R} = \mathbf{C}^{\dagger}\mathbf{F} \tag{4.26}$$

Now the stability of the P controller for the reaction wheel must be proven in the same way as the thruster PD controller. The input to the controller is the desired angular velocity of the target simulator. The control torque is calculated by multiplying the error between the desired angular velocity and the current angular velocity by the proportional gain K_p. This is then used to compute the angular acceleration of the reaction wheel. Integrating this we get the angular velocity of the reaction wheel. Then the angular momentum of the reaction wheel is transferred to the target simulator by the conservation of angular momentum. The full process is shown in Figure 4.6.

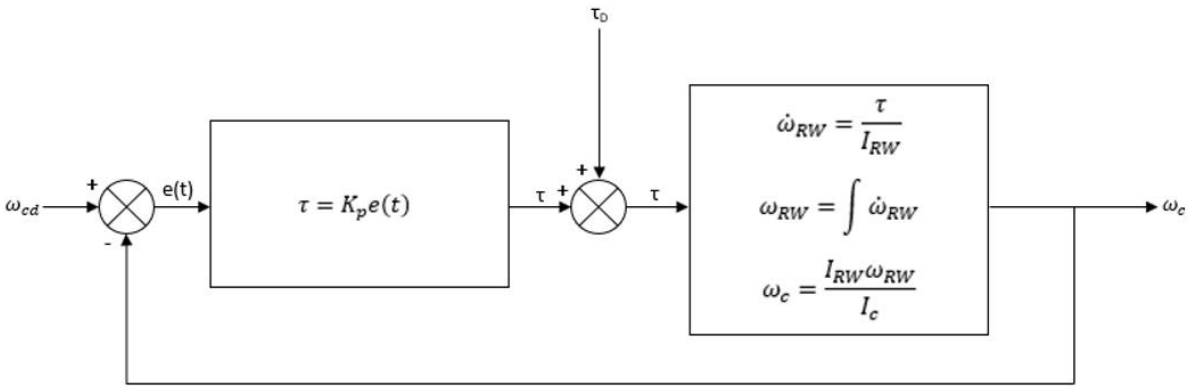


Figure 4.6 P controller used to maintain angular velocity of the target simulator.

The controller is then changed to the s-plane to determine its stability. Closing the loop of the control system, we obtain the closed loop transfer function, see Figure 4.7. This first-order system contains only a single pole and no zeros.

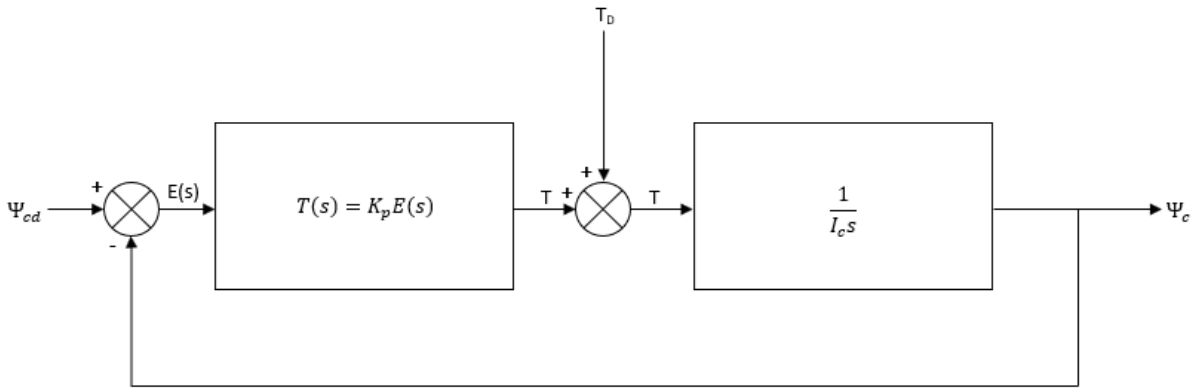


Figure 4.7 P controller represented in the Laplace domain to check stability.

$$T(s) = \frac{\frac{K_p}{I_c}}{s + \frac{K_p}{I_c}} \tag{4.27}$$

To assess the stability of the transfer function, the denominator of the transfer function is assessed. By using the Routh-Hurwitz criterion again, we can assess the stability of the controller. For the system to be stable there cannot be any sign changes in the n^{th} column. Therefore, the gain value of the K_p term must be positive to be stable.

$$D(s) = s + \frac{K_p}{I_c} \tag{4.28}$$

Table 4.2 Routh table to determine the stability of the P controller

Sign Change	Order of Term	n	n-2
	s^1	1	0
+	s^0	K_p	

The settling time of the system is inversely proportional to the controller gain. This implies that a gain of infinity would result in a settling time of zero. However, this is not possible due to the limitations of the reaction wheel. The reaction wheel on the target robotic simulator is a Picosatellite Reaction Wheel RW-0.01 from Sinclair Interplanetary based in Toronto, Canada. The maximum angular momentum storage of the wheel is 0.01 Nms, which is too small for the simulator. It has a limited operation range and can easily be saturated. To determine the gain for the controller we asses the maximum angular momentum the reaction wheel can store, as well as the maximum torque the controller can output. The required datasheet values are listed below.

Table 4.3 Datasheet values for Picosatellite Reaction wheel RW-0.01

Maximum Angular Momentum	0.018 Nms
Maximum Torque	0.01 mNm
Mass	120 g
Dimensions	50 mm x 50 mm x 30 mm

Using the known dimensions and mass of the reaction wheel the maximum angular velocity of the reaction wheel can be determined from the maximum angular momentum from the datasheet. The maximum angular velocity of the reaction wheel is 60 rad/s.

$$L_{RW\max} = I_{RW}\omega_{RW\max} \tag{4.29}$$

$$I_{RW} = \frac{1}{2}M_{RW}r^2 = \frac{1}{2}(0.120g)(0.05)^2 = 0.0003kgm^2$$

$$0.018Nms = 0.0003kgm^2\omega_{RW\max}$$

$$\omega_{RW\max} = 60rad/s$$

We then transfer this to the maximum angular velocity that this can induce on the target simulator. Using the maximum torque that the reaction wheel can apply and the maximum angular velocity the gain K_p can be determined. The simulator will begin at zero angular velocity to start and will have to reach the target angular velocity. The maximum change that can occur in the angular velocity is starting from zero to ω_{Cmax} as the target simulator will only tumble in a single direction. The gain for the reaction wheel becomes $K_p = 0.025$. The resultant pole of the system can be seen in Figure 4.8.

$$L_{RW\max} = I_{RW}\omega_{RW\max} = I_c\omega_{c\max} \tag{4.30}$$

$$\omega_{C\max} = \frac{I_{RW}}{I_C}\omega_{RW\max} = \frac{0.0003kgm^2}{0.456kgm^2}60rad/s = 0.039rad/s$$

$$\tau = K_p\omega_{c\max} \tag{4.31}$$

$$0.001Nm = K_p * 0.0394rad/s$$

$$K_p = 0.025 \tag{4.32}$$

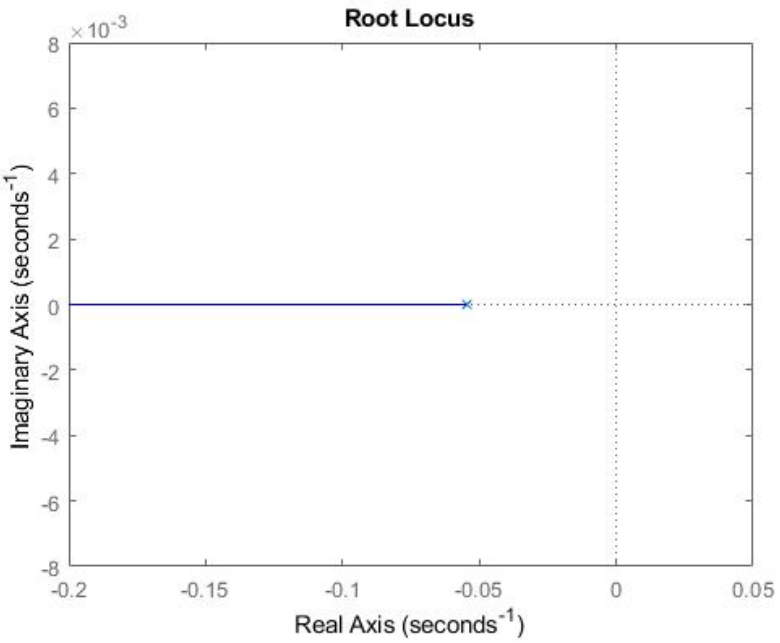


Figure 4.8 Root-Locus plot for P controller for target simulators angular velocity.

4.3 Adaptive Control

The challenge with the on-off thrusters is the fact that the same set of thrusters are used for both the translational and rotational components of the system. When a thruster is activated to correct the translational component, it will apply a torque onto the system causing it to rotate. This occurs for two reasons; The first is due to the center of mass of the simulator not being exactly located at its geometrical centre in the *xy* plane, while the thrusters are located symmetrically about the geometrical center in the xy plane. The

second is due to the placement of the thrusters on the simulator not being directly on the center of mass of the simulator. If the thrusters were directly on the center of mass, the thrusters would not be capable of contributing to the rotation of the simulator.

This causes an activated thruster to correct the rotational component of the simulator and applies force to translate the simulator. The simulator constantly oscillates between correcting translational and rotational motions. When this occurs, the simulator will actuate thrusters to move in the travel direction less often as the same thrusters are used to correct the simulator's position and orientation. This oscillation increases the traveled path length and can also cause the simulator to get stuck at a desired waypoint as it will constantly attempt to correct the attitude of the simulator but would oscillate around the desired orientation. These delays cause the simulator to take a longer time to travel to a desired location and thus consume more fuel. An illustration of this is depicted in Figure 4.9, where the two thrusters are only used to move the simulator along the travel direction at step 2.

Also, due to the nature of the PD-controller using the on-off thrusters there is a minimum distance from a desired waypoint where the thrusters will not activate. Within this distance, the controller cannot activate the thrusters to correct errors as the errors are too small. The simulator will drift until it reaches a point where the error is sufficiently large again and then actuate the thrusters. This is illustrated in Figure 4.9 step 2-3.

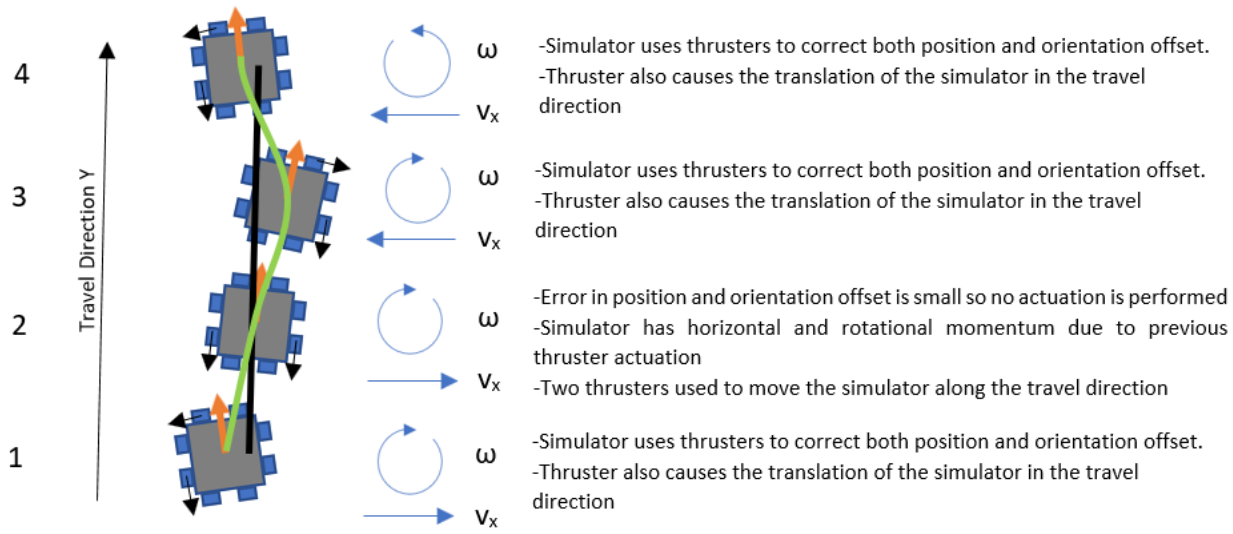


Figure 4.9 Illustration of errors due to on-off thruster actuation with PD-controller.

An improved controller is needed to prevent the large oscillations about the waypoints by maintaining the position and orientation of the state components while still correcting for errors.

The challenge for this controller is that it must be computationally efficient so that it can be implemented on the onboard computer of the simulator. To do this, a simple adaptive controller is developed where the PD controller is enhanced by using Fuzzy Gain Scheduling to improve its performance.

"Fuzzy logic is used to model logical reasoning with imprecise statements" [95]. It is a generalization of standard logic. It involves the degree in which a category or condition is met. It represents this degree between 0 (completely false) and 1 (completely true) [96]. An example would be determining someone's height. The categories could be short, average height, and tall. Then we can say that, 'John is short', with a degree of truth of 0.9,

'John is average height' with a degree of truth of 0.1, and 'John is tall' with a degree of truth of 0. This can then be used to map these values to a required output. "Fuzzy logic offers a convenient way to map an input space to an output space" [97].

The fuzzy logic begins by breaking down the problem into a fuzzification step, which maps the inputs into a linguistic set. These are then fed into a rule evaluation and produces a fuzzy output. Then a de-fuzzification procedure is required to convert the fuzzy output into a nonfuzzy control action [98].

The fuzzy logic model implemented here is a single input first-order Sugeno fuzzy model. This model allows for the mapping to a crisp output through a weighted sum. Using a weighted sum reduces the computation required by avoiding the defuzzification process [99]. An example of a single-input first-order Sugeno fuzzy model is:

$$\begin{cases} If \ \text{x is } A_1 \ \text{then } f_1(x) \\ If \ \text{x is } A_2 \ \text{then } f_2(x) \\ If \ \text{x is } A_3 \ \text{then } f_3(x) \end{cases}$$

Then using the output of the above rules, they are then weighted and summed to provide a crisp output.

4.3.1 Fuzzy Logic Implementation

For the adaptive controller, we will use one input value of the simulator position as the input of the fuzzy logic. The derivative gain of the controller will also be a function of the position of the simulator. If the simulator is close to the desired position, we would like

the velocity to remain close to zero. As the fuzzy logic output must be continuous, the output of the system is the gains for the respective P and D components of the controller. These gains are then fed into the PD-controller. The gains K_p and K_v are functions of the position and orientation error.

$$\mathbf{F} = \mathbf{M}(\mathbf{K}_{\mathbf{p}}(\mathbf{e}(t))\mathbf{e}(t) + \mathbf{K}_{\mathbf{v}}(\mathbf{e}(t))\dot{\mathbf{e}}(t)) \tag{4.33}$$

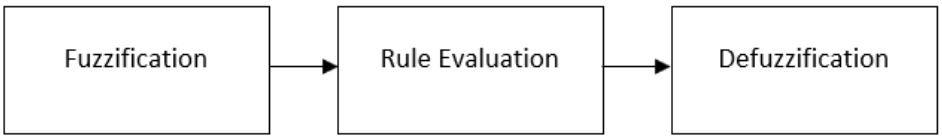


Figure 4.10 Steps for developing a fuzzy logic controller.

There are three steps in implementing fuzzy logic as shown in Figure 4.10. The first is the fuzzification where the inputs are mapped to membership functions. Then the membership functions are mapped to output membership functions using a set of rules and then a defuzzification is performed to provide a continuous output.

First, the fuzzification is performed. There are two regions of interest in the position and orientation space regarding the PD controller and the path planning way pointing. The first is when the simulator position or orientation error is greater than the waypoint switching distance and the second is when the error is within the waypoint switching distance. This is mirrored for positive and negative numbers, see Figure 4.11. With this we are then left with three regions of interest, Negative-PD, Under-Actuated-Region, and Positive-PD.

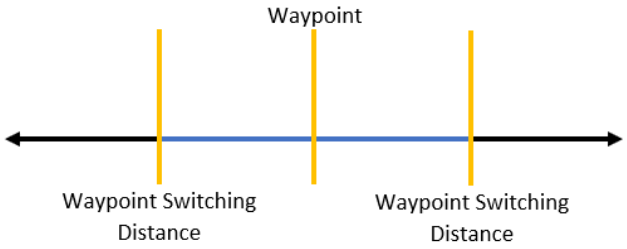


Figure 4.11 Illustration of waypoint switching distance from the waypoint.

Next step is the rule mapping and the design of the membership functions. It is common to use triangular membership functions to specify regions of interest and trapezoidal membership functions for boundaries. Here a triangular membership function is centered on the waypoint. This represents the region within the waypoint switching distance where the standard PD-controller would not actuate any of the thrusters. At the waypoint distance a trapezoidal membership function is used to represent the boundary of the PD controller. This is reflected in both the positive and negative regions.

$$\begin{cases} If \ \text{e(t) is Negative-PD then } MF_1 \\ If \ \text{e(t) is Under-Actuated-Region then } MF_2 \\ If \ \text{e(t) is Postive-PD then } MF_3 \end{cases}$$

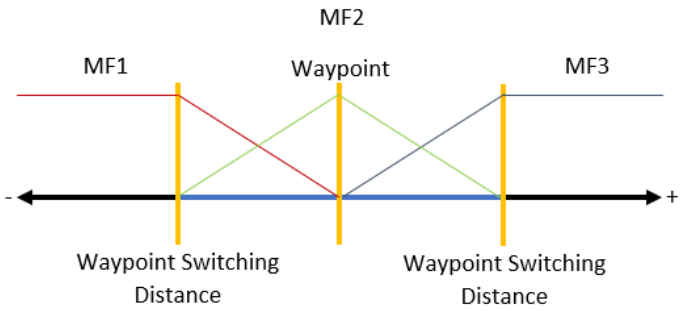


Figure 4.12 Illustration of membership functions over waypoint domain.

The final step is to perform the defuzzification process by converting these rules into a discrete output. For the Sugeno Fuzzy model this is done by computing a weighted sum of all the membership functions is performed as it is a common method for defuzzification. This performs a single gain value that can be used for K_p and K_v. The weighing for membership function 1 and membership function 2 is to use the standard gains of the original PD controller. The weighing for MF_2 is to use a larger gain to hold on the desired waypoint.

A sample of the weighted sum of the membership functions is shown below to compute the K_p term, see Figure 4.13. Here, p is the measured position away from the desired waypoint. Membership function 1 has a weighting of 0.8 of its input of point p. Membership function 2 has a weighting of 0.2 of its input of point p. The output is a combination of the high gain to maintain the system at the waypoint and the standard PD gain.

The gains for membership functions 1 and 3 will be the same as that of the gains computed in the PD control strategy. The gains for membership function 2 was determined

through brute force using the existing gains of the PD controller as a starting point. The gains were increased until there was no increase in performance with an additional increase in gain.

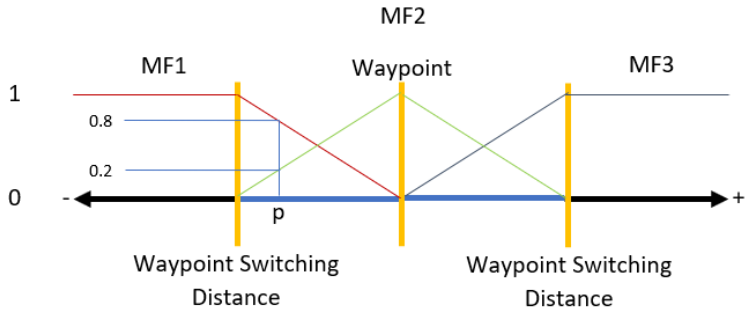


Figure 4.13 Illustration of weights from activated membership functions.

Table 4.4 Sample calculation of weighted summed from fuzzy logic gain scheduling

Membership Function	Output Weighting	Input Weight		
MF1	0.5	0.8	0.5 X 0.8 =	0.4
MF2	1.5	0.2	1.5 X 0.2 =	0.3
MF3	0.5	0.0	0.5 X 0 =	0
			K_p =	0.7

The goal of the adaptive controller is not to improve the accuracy of following the waypoint path but to prevent the system from oscillating about a waypoint. The system will attempt to correct overshoot and control the simulator in the unactuated region of the PD controller alone. This will cause the simulator to take an overall shorter path but cause the

simulator to have a higher frequency oscillation in the attitude control. These smaller oscillations keep the overall heading of the simulator along the path allowing for two thrusters to be used to accelerate the controller along the straight-line path, see Figure 4.14.

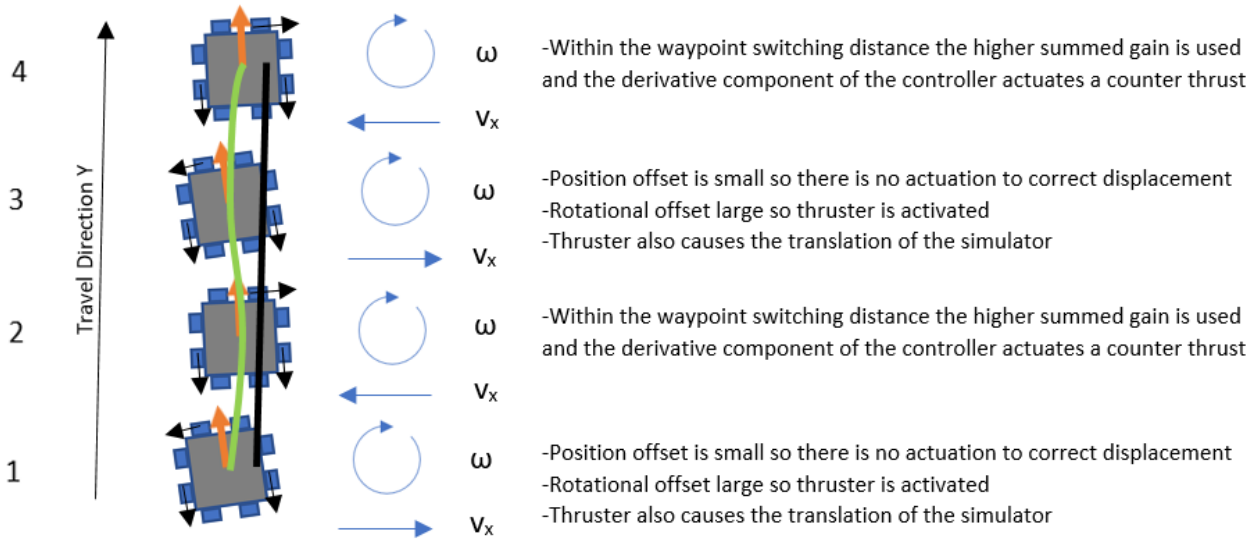


Figure 4.14 Illustration of path taken by simulator with fuzzy logic gain scheduling.

The adaptive controller using fuzzy gain scheduling designed here will have the highest performance improvement for linear paths as one component of its pose will always be maintained. As the order of the path planning algorithm increases the performance improvement of the adaptive controller reduces. In the worst-case scenario, the path to be followed is circular where all components of the simulator's position and orientation are always changing. In the designed path planning algorithm, the best case while still traversing a path (linear path) and worst case (circular path) are demonstrated.

To further explain how the circular path is one of the worst-case scenarios, we look at how the controller operates. The goal of the controller is when one or more of the states remains constant along a path it will better hold that state to prevent oscillations. As with the explanation of the linear path earlier, the orientation state variable would remain constant. With the increased gain, see Figure 4.15 and Figure 4.16, it will better hold its orientation. In Figure 4.16, the gold dot represents the error from the desired waypoint, where the point with the vertical green line is the point with zero error.

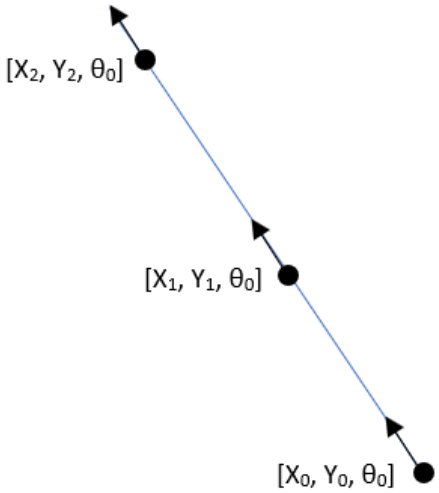


Figure 4.15 Linear path with orientation state remaining constant.

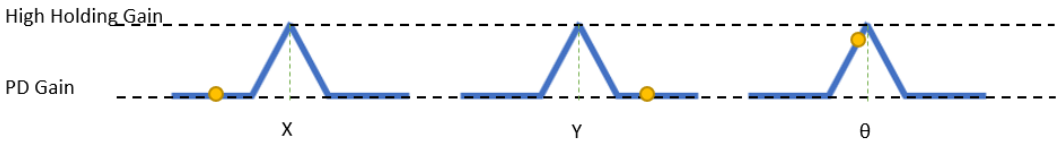


Figure 4.16 Sample gain structure for linear path.

This control structure would also show improvement for a stationary, rotating structure, see Figure 4.17. This manoeuvre is not used in this work and has no value for the chaser's motion but is used here to illustrate the controller's function. Here, the position remains constant with the orientation constantly changing. This is represented by Figure 4.18 where the X and Y portions of the controller have higher gains, while the orientation operates in the PD region.

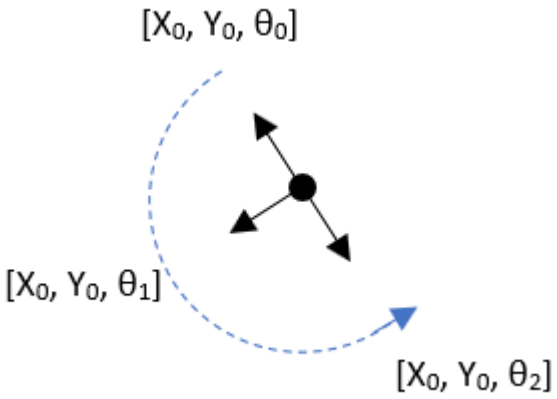


Figure 4.17 Hold in place and rotate manoeuvre.

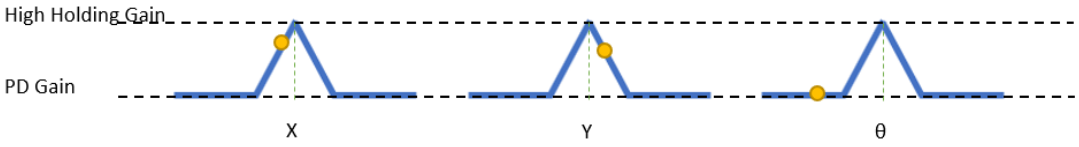


Figure 4.18 Sample gain structure for hold in place and rotate manoeuvre.

With the circular path, while maintaining its orientation towards the center of the circular path, all the state variables are constantly changing. The controller would operate as the

standard PD controller. Any path where all three state variables are constantly changing is the worst case for this controller's operation. The path used here is one of the worst cases as the benefits of the high holding gain would be unused. The goal of the controller was to have an improvement over the standard PD controller. With no increased benefit from using the adaptive controller it is one of the worst cases for this controller.

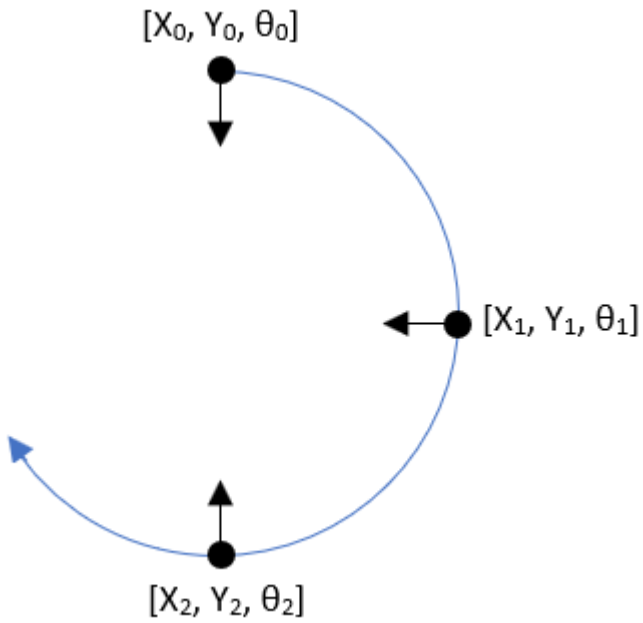


Figure 4.19 Circular path with orientation pointing towards circular path's centre.

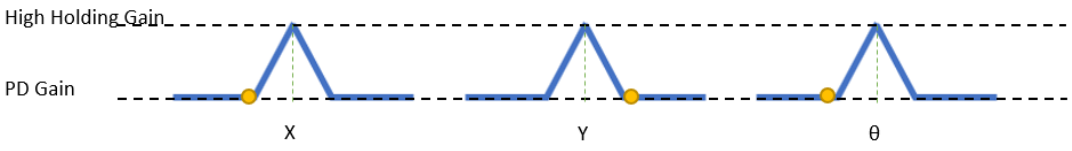


Figure 4.20 Sample gain structure for circular path with orientation pointing towards circular path's centre.

Chapter 5 SIMULATION SOFTWARE DEVELOPMENT AND TUNING

Summary: This chapter includes three sections. The first section describes why the simulation was created and what are its limitations. The second section is the development of the Matlab, and Simulink simulation models developed to test path planning and control algorithms. The results for the simulator are shown and then tuned to further optimize the system. The third section is the modification of the simulation for the addition of gain scheduling using the fuzzy logic controller to create a simple adaptive controller.

5.1 Simulation Implementation

Testing and adjusting the software on the robotic simulator was quite a difficult task as the air supply tanks could only store a limited amount of air. Currently, the fully charged air tanks can only support about 10 minutes of testing. Then the tanks require re-charging which take approximately 20 minutes for both simulators. This limits the number of tests that can be performed in a day. Another limiting factor was the battery on the simulator. The battery of the simulators only holds a charge for approximately a day

without needing to be re-charged. The battery requires a full day to charge and only one simulator can be charged at a time with the current setup.

The purpose of developing the simulator was to assist in the design of the hardware simulator, its algorithms. Using the simulation allowed for algorithms to be quickly developed and test various path planning and control algorithms. This reduced the total development time for algorithms due to the limitations of the air supply and battery on the physical simulator. In addition, the simulation was used to test the calculated stability and gains for the PD controller.

A Matlab and Simulink simulation was developed to do this. The simulator used the Simscape Multibody Library to create a dynamically similar simulation. The immediate benefit from developing this system was to test the path planning algorithm for the rotating target case. The original path planning algorithm did not predict waypoints ahead so the simulator can catch up to the target face. This caused the chaser simulator to constantly lag the target simulators rotation rate. By projecting the waypoints forward by the measured rotation rate the chaser simulator can catch up with the target and then synchronize with the target.

The simulator was also used to test the gains for the system. The original gains for the system were determined numerically. The computed gains were computed at the thruster activation threshold and served as the minimum gain for the system to operate. After running the simulation, this left the system to be underactuated. Using the simulation, quick and repetitive testing. increasing the gains using brute force to determine the gains that would prevent under actuation. Using the physical hardware would have taken several

days to determine the required gains. Using the simulation allowed for the quick repetition of tests, allowing for the gains to be determined in hours.

The limitations of the simulation are that its inertia properties are approximated. The base simulator and the manipulator links are approximated as solids with a uniform mass distribution, which is not the case of the physical hardware. In addition, the testbed environmental conditions are not accurately simulated. The testbed operates in a microgravity environment, where as the simulation operates in perfect zero gravity. The testbed also suffers from dust on its surface. This causes additional disturbances that are not emulated in the simulation. Finally, the simulated sensors are perfect, not exhibiting the noise that the actual hardware would experience.

5.2 PD Controller Simulink Software

The Simscape Multibody Library was used to develop the simulated physical components of the robotic simulators and to dynamically couple the bodies. All simulated sensors were programmed to have the same update frequencies measured from the developed sensor hardware making the simulation having the same sampling limitations as the real system, see Figure 5.1.

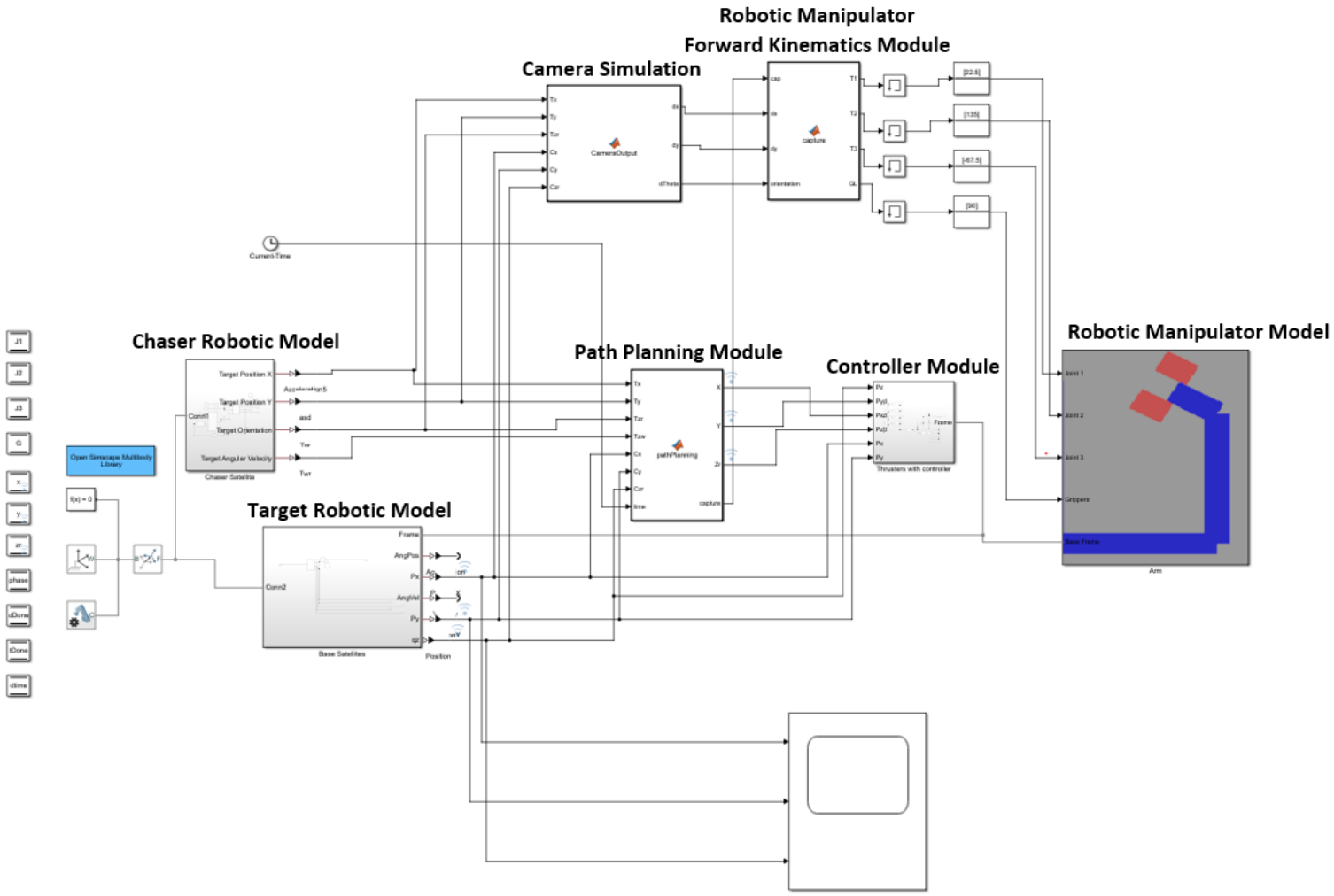


Figure 5.1 Simulink simulation for target and chaser simulators.

5.2.1 Target Robotic Model

This module simulates the rotating target simulator. It consists of two dynamic bodies, the Target Base with the attached Target Capture Point, see Figure 5.2. The two bodies are rigidly fixed to each other. The Target Base is attached to a 3 degree of freedom module, allowing it to rotate and translate freely. The inputs for this module are the desired T_x and T_y target position in space as well as the Target Path Planning module that uses the desired input spin rate and the current position of the target robotic simulator to update the simulator to the next position. The outputs of the system are the X, Y position of the base simulator as well as the target's orientation and angular velocity in the world frame. The input for this module is the world frame coordinates.

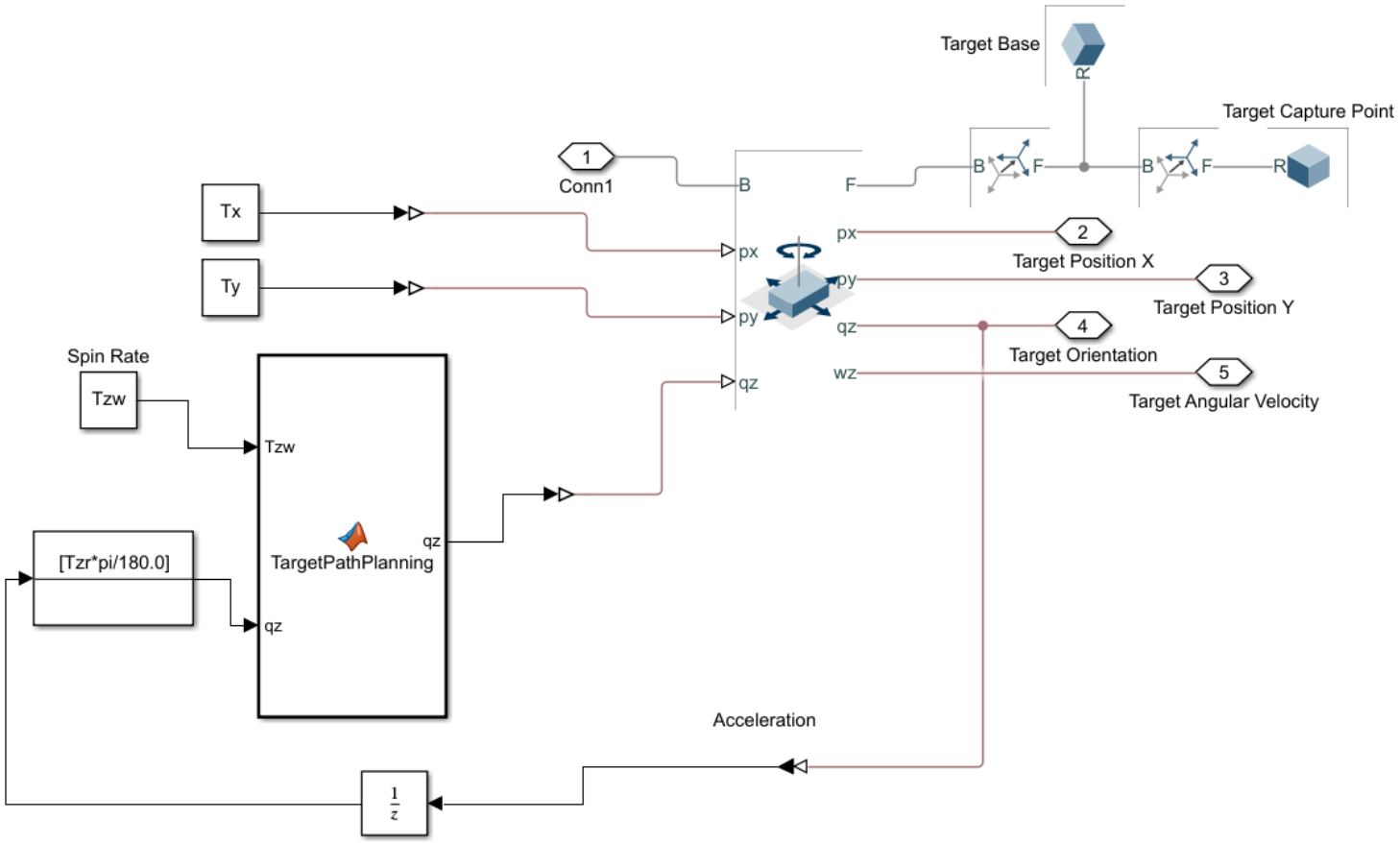


Figure 5.2 Simulink simulation of target simulator.

5.2.2 Chaser Robotic Model

The chaser robotic model consists of a 3 degree of freedom module, allowing it to rotate and translate freely with the base dynamic body attached to it. This allows the base dynamic body to rotate and translate freely in the plane. The input for this module is the simulations world frame. The outputs for the system are the chaser simulator's reference frame where the robotic manipulator will be attached to. The angular position, velocity and acceleration of the chaser base are outputted. The X and Y position of the chaser simulator is outputted.

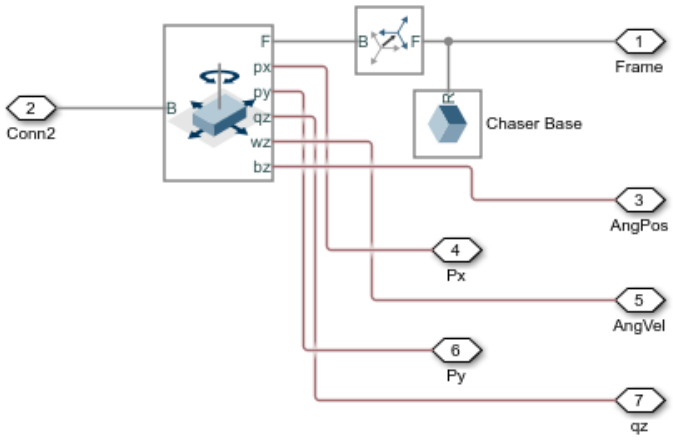


Figure 5.3 Simulink simulation of chaser simulator base.

5.2.3 Path Planning Module

The path planning module determines the way pointing and maneuvering of the chaser simulator. It also performs the check if the chaser simulator is in the capture

position. The algorithm used here is the same as described in the Path Planning chapter. The inputs for the Path Planning Module are the target simulator's (X, Y) coordinates, orientation and angular velocity, the chaser simulator's (X, Y) coordinates, orientation and the system time. The outputs of this module are the next (X, Y) coordinates and orientation waypoint that the chaser simulator must reach.

5.2.4 Camera Simulation Module

This module converts the position and orientation data of the target and chaser system, so that it corresponds with the data received from the camera and then converts the camera frame data to correspond with the targets capture point. This would output values from a camera placed on the side of the chaser simulator to the target capture point. This module is required as the camera is how the chaser will determine the targets position and orientation in the world frame. This is only used for the Robotic Manipulator Forward Kinematics Module as the path planning and positioning system would require it to convert back to world frame system. The inputs to the system are the target simulators X, Y, and orientation, and the chaser simulators X, Y and orientation in the world frame. The output of the system are the X, Y and orientation of the target simulator with respect to the camera frame attached on the center of the positive X face on the chaser simulator.

5.2.5 Robotic Manipulator Inverse Kinematics Module

This model checks to see if the manipulator capture requirements set from the path planning module have been satisfied. If the chaser is in the capture position, then the forward kinematics of the manipulator is computed using the known joint angles. If the end

effector of the manipulator reaches the capture point the gripper will close. The next waypoint of the manipulator is computed and the inverse kinematics for the next waypoint is determined. If a solution for the manipulator exists, then the manipulator motors are actuated. The inputs to this module are the X, Y and orientation of the target simulator capture point and the capture command from the Path Planning Module. The outputs of the system are the new joint angles for the simulated robotic manipulator and the gripper finger angles.

5.2.6 Controller Module

The controller module simulates a PD-controller and the thrusters for the chaser simulator. The PD-control and thruster allocation are computed as the PD-control section of Chapter 4.1 PD Control Strategy. Four thruster dynamic blocks are used to simulate on-off bi-directional thrusters with a single force value. The inputs of this block are the actual and desired X, Y and orientation for the chaser simulator. The output is the chaser simulator's reference frame so the thrusters can be dynamically coupled to the chaser simulator, see Figure 5.4.

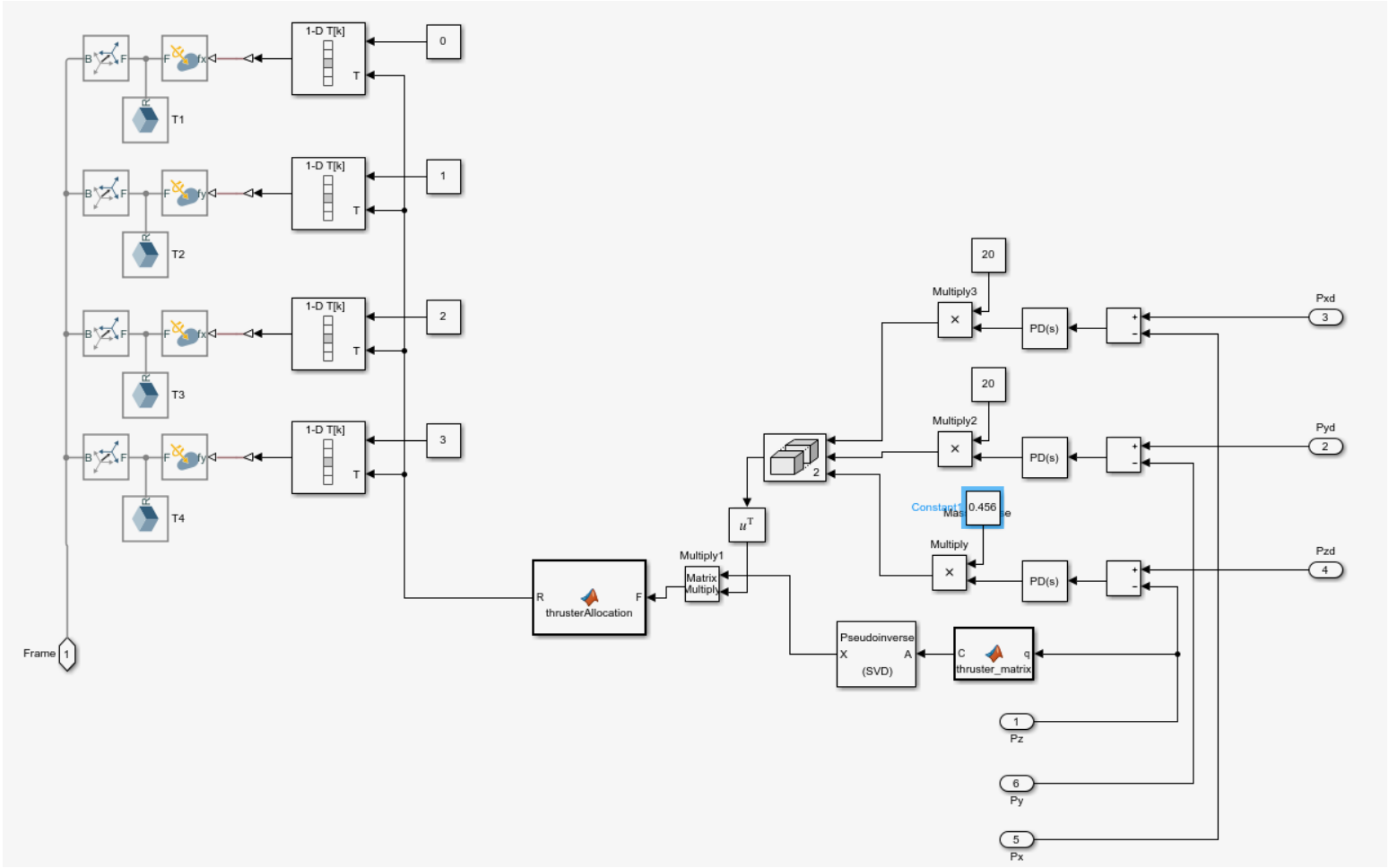


Figure 5.4 PD controller and thruster allocation.

5.2.7 Robotic Manipulator Model

This model simulates and actuates the robotic manipulator. It consists of a 3-DOF manipulator with a two-finger gripper. The input of the system is the chaser simulator's reference frame so that the system is dynamically coupled to the base frame. The three joint angles and the gripper angle are inputted into the system. Three rotational joints are used to actuate the system, see Figure 5.5.

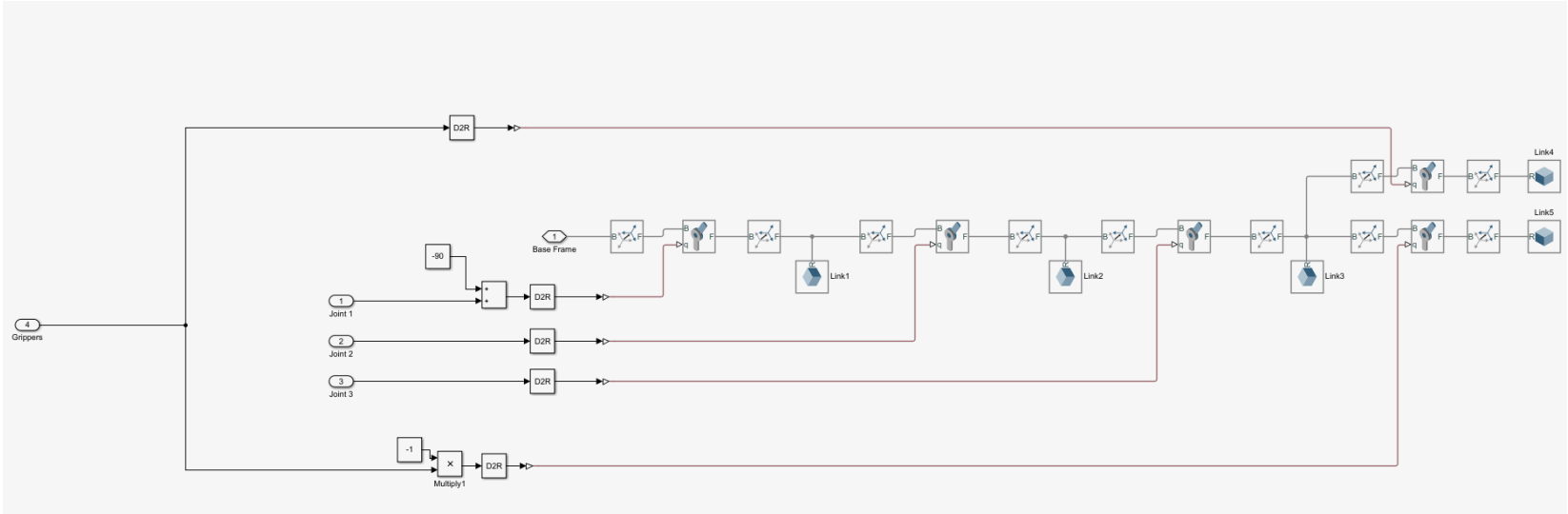


Figure 5.5 Simulink simulation of robotic manipulator on chaser simulator.

5.2.8 Simulation Results and Tuning

The results from the simulation (see Figure 5.6) use the gains computed from the PD control determined in Chapter 4.1 PD Control Strategy. As shown in the output results Figure 5.7-Figure 5.10, there is a clear variation in the path from the way-pointing path. When switching from the straight-line path to the circular path, there is an overshoot in the X direction from the straight-line path and the simulator gains are too low to initiate the circular maneuver. The simulator slowly drifts away from the waypoint until the thrusters can be activated again. Although the system is critically damped mathematically, the on-off nature of the thrusters causes the gains to deviate from the mathematical model.

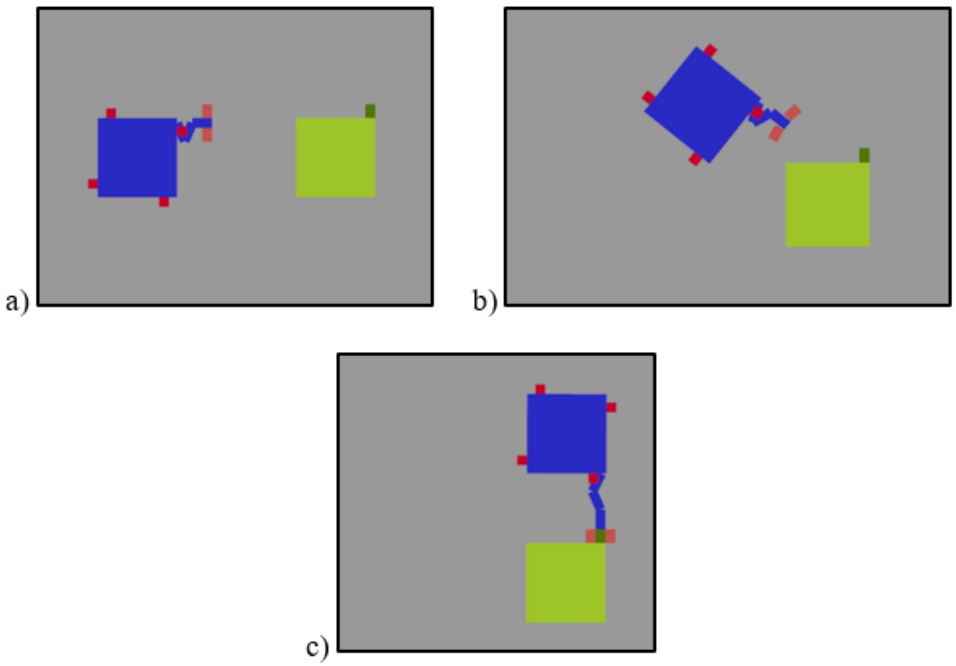


Figure 5.6 Output of Simulink simulation a) Chaser simulator approach b) Chaser simulator synchronization c) Chaser simulator capture of target simulator.

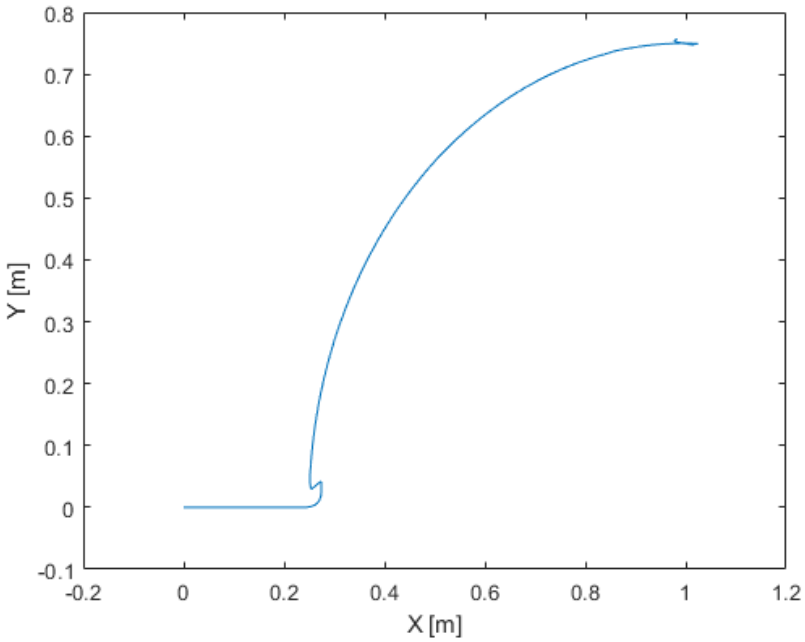


Figure 5.7 Potion of the Simulink robotic simulator.

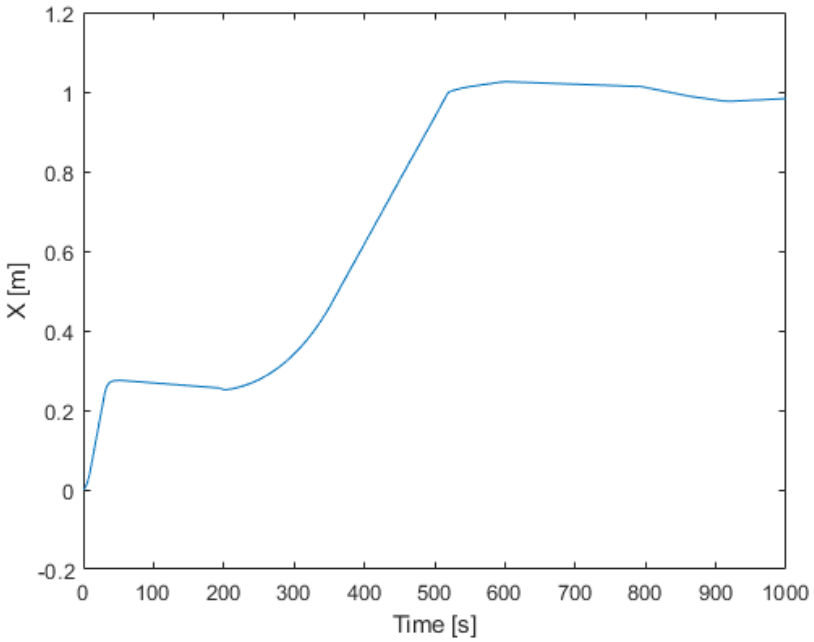


Figure 5.8 X position of the chaser simulator.

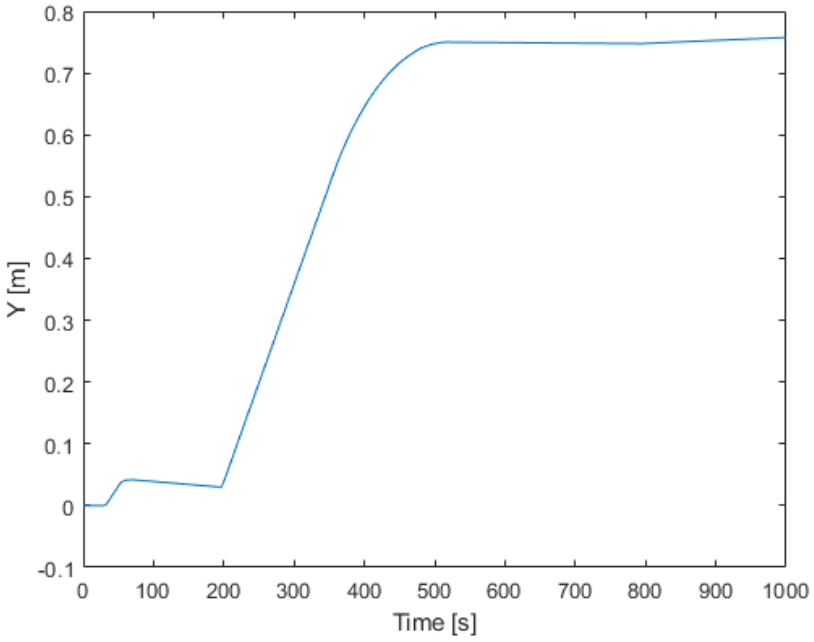


Figure 5.9 Y position of the chaser simulator.

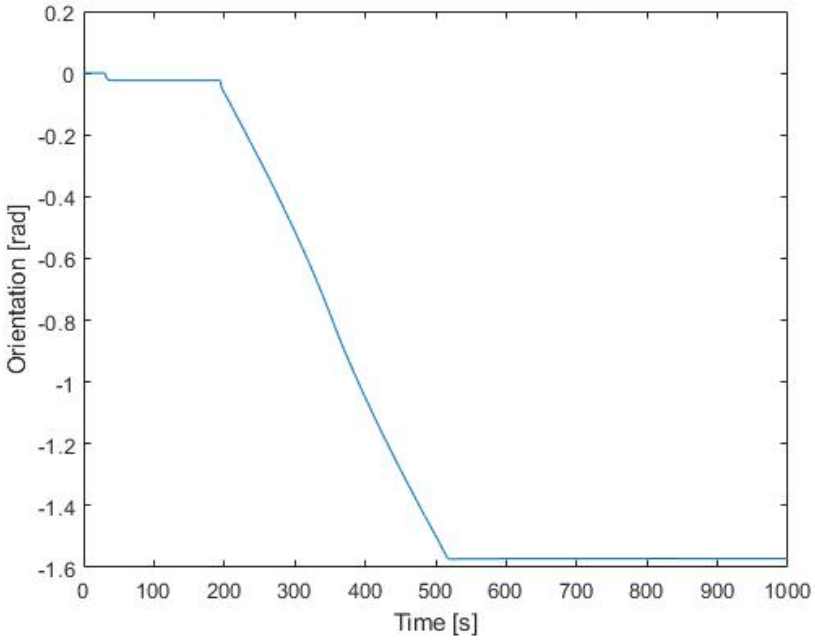


Figure 5.10 Orientation of the chaser simulator.

To correct this, the gains for the controller are increased slightly to initiate the thrusters more readily to remove the overshoot. The new gains used for the system are $K_p = 0.38$ and $K_v = 1.8$ for the translational component of the system. The gains for the controller responsible for rotating required no adjustment from the theoretical results.

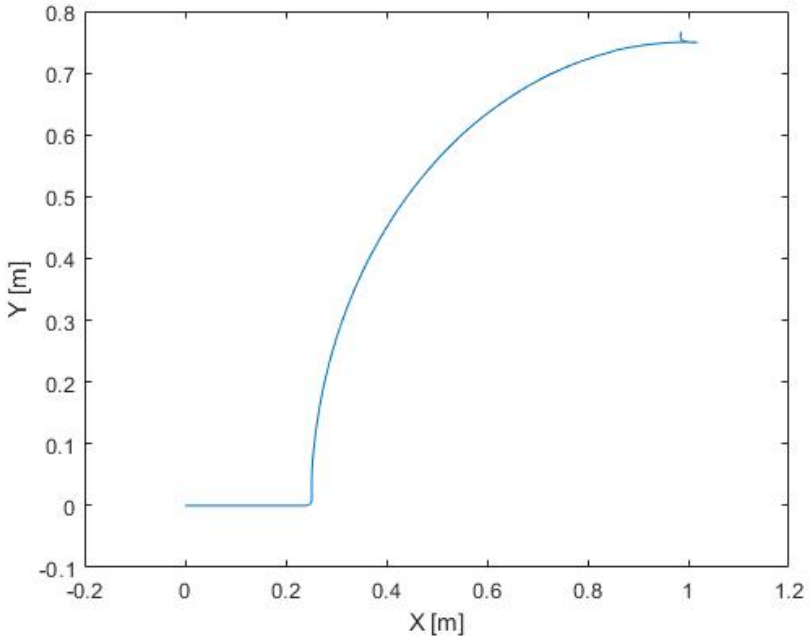


Figure 5.11 Y Position of the chaser simulator after gain correction.

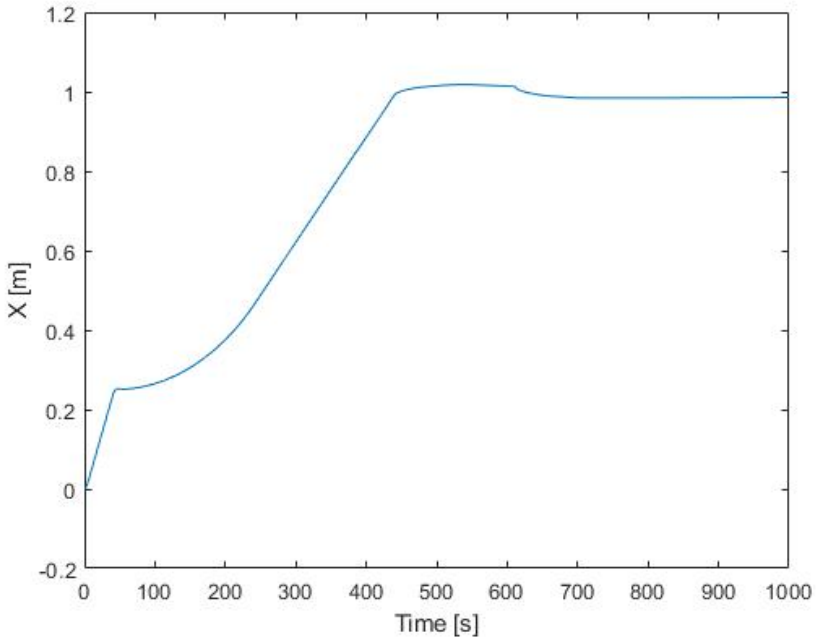


Figure 5.12 X position of the Simulink chaser simulator after gain correction.

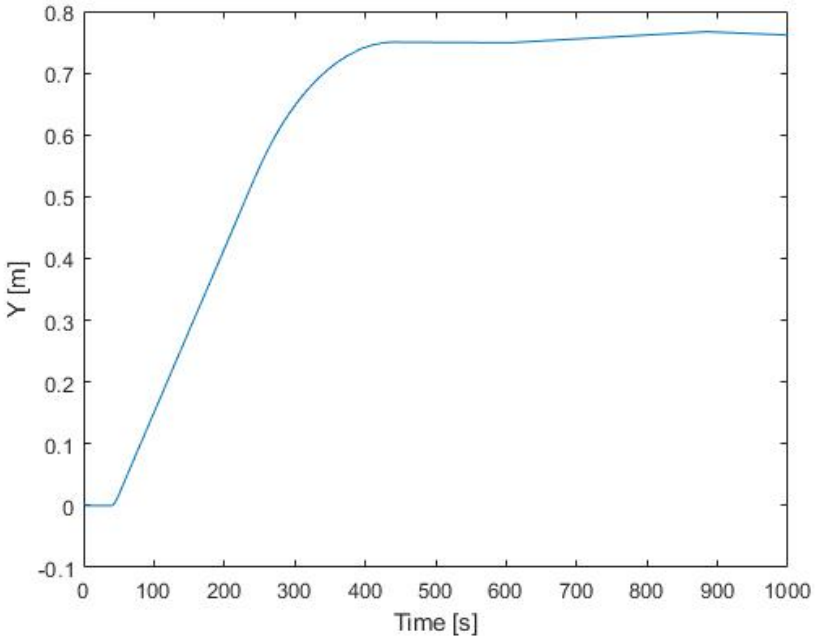


Figure 5.13 Y position of the Simulink chaser after gain correction.

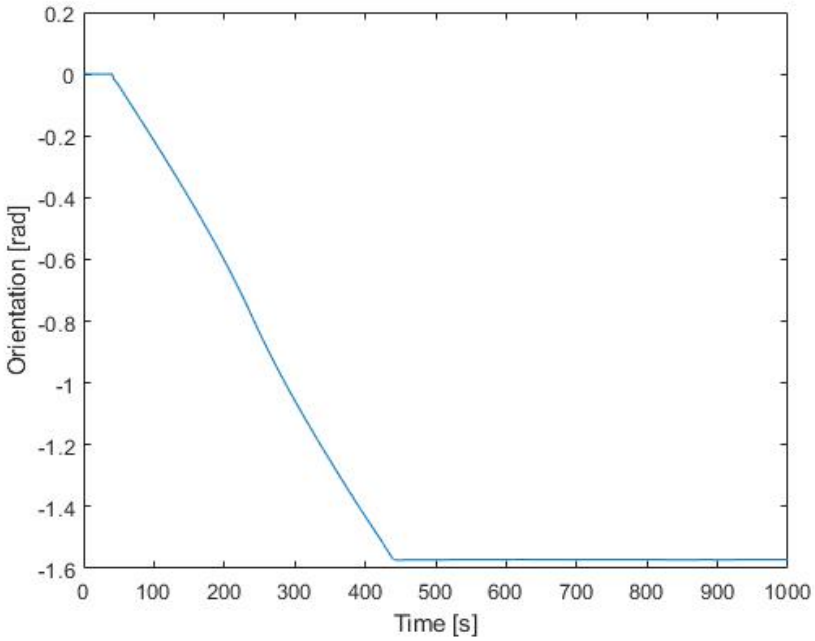


Figure 5.14 Orientation of the chaser simulator after gain correction.

5.3 Adaptive Control Simulation Software

The simulation software for the adaptive controller using fuzzy gain scheduling is the same as shown for the PD-controller except for the gains. In the PD controller version, constant values were used for the gains. Unlike the PD controller, fuzzy logic is used to provide the gains. The input from the fuzzy logic module is the difference in the measured and desired values.

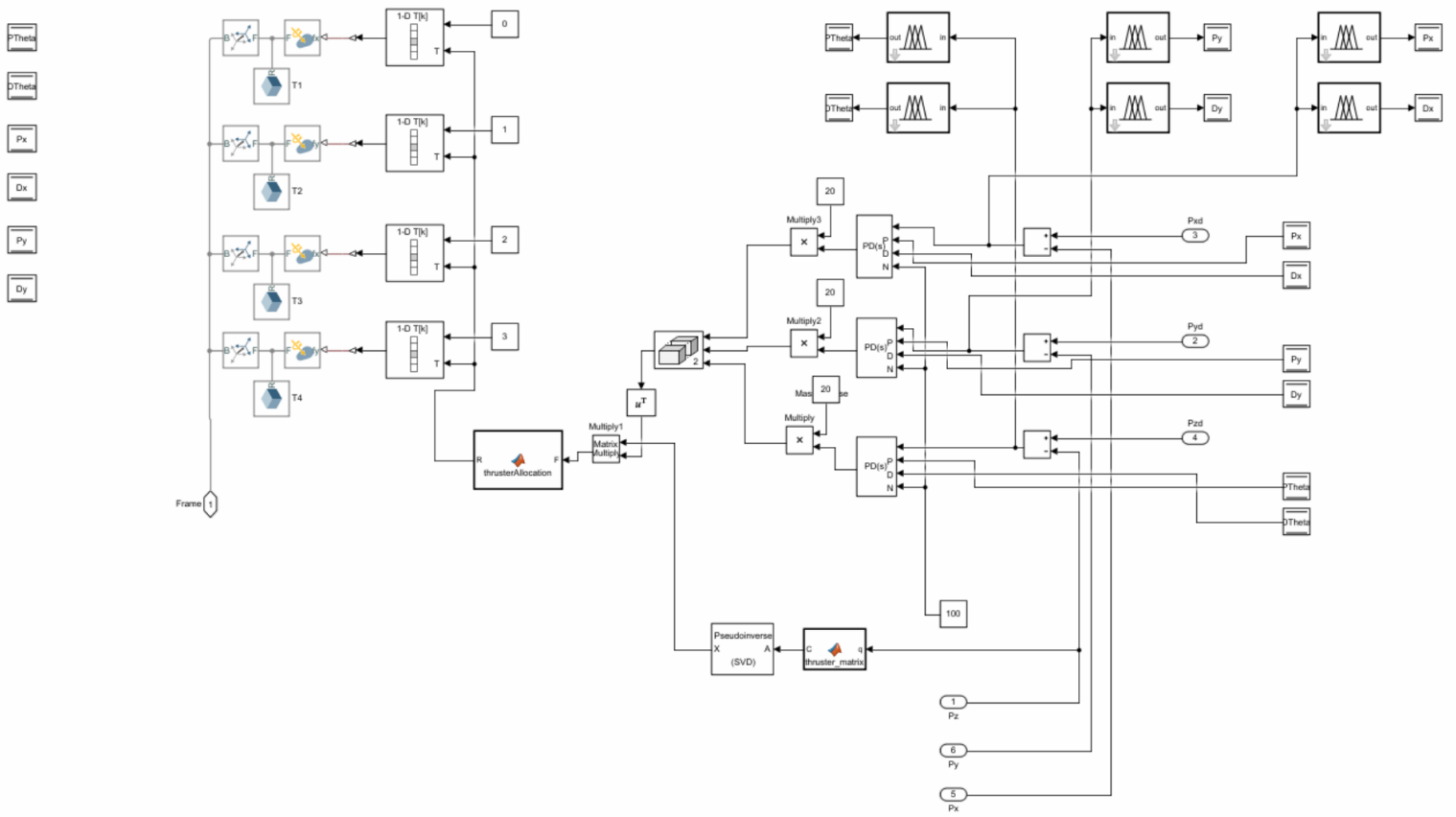


Figure 5.15 Simulation modification for adaptive controller.

The gains for the membership functions 1 and 3 were kept the same as the gains from the PD controller. The gains for the membership function 2 for the controllers were determined using trail and error beginning with the values of the PD controller and slowly increasing equally between translational and rotational components until the change in the membership function gain was negligible. The final gains used for the membership function 2 were $K_p = 0.58$ and $K_v = 3.6$ for the translational controllers. For the orientation controller the gains for membership $K_p = 2.51$ and $K_v = 6.22$. This represents a 0.2 gain in the proportional value and a doubling of the derivative component of the controller.

Chapter 6 CAPTURE RESULTS

Summary: This chapter contains two main sections. The first section is the capture using the PD-controller for three capture senecios: stationary controlled, stationary free-floating and tumbling targets. The second section has the PD controller replaced by an adaptive controller. For the same initial conditions both controllers are compared.

6.1 PD Controller Capture Results

Implementing the same PD controller in the simulation on the physical simulator, we perform the capture for 3 scenarios. The same adjusted gains from the simulation were used, they were varied to determine if the performance could be increased. Due to the large noise in the measurements the variation of the gains did not improve the performance of the system. Using the PD controller, three capture scenarios are tested. The first is the stationary capture of a controlled target simulator. The second, is the capture of a stationary free-floating target and the final is the capture of a tumbling free-floating target.

6.1.1 Stationary Controlled Target

The first is the capture of a stationary controlled target, where the thrusters and reaction wheel are used to control the target simulator with a specific orientation and position. When the robotic manipulator contacts with the target simulator, it will place forces on the target that can be considered disturbances. The target simulator will react and compensate for these disturbances to hold the desired position and attitude.

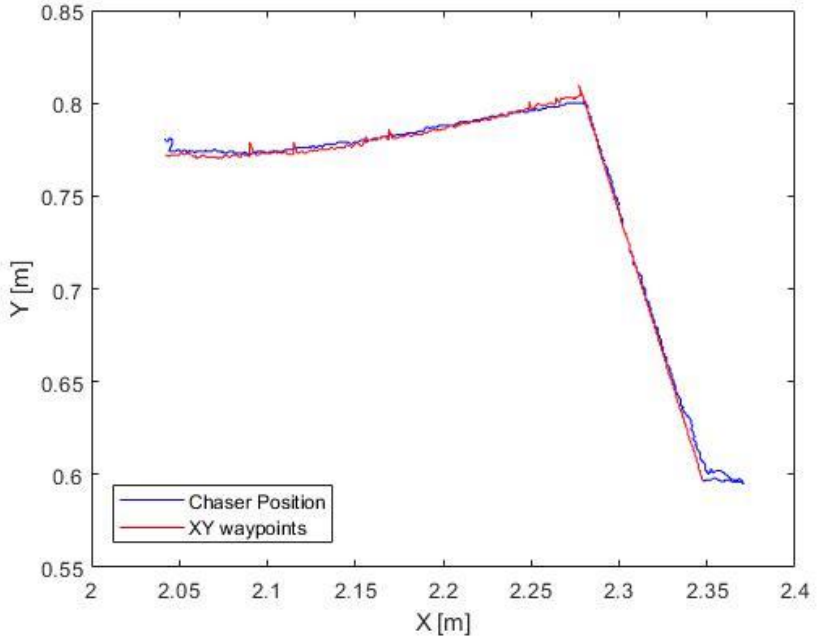


Figure 6.1 Position data of capture of controlled target simulator.

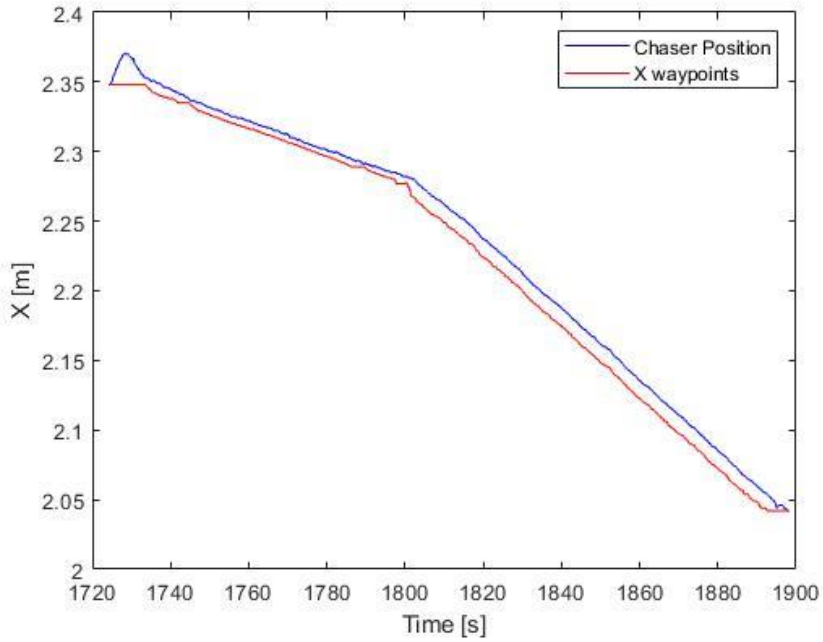


Figure 6.2 X position data of capture of controlled target.

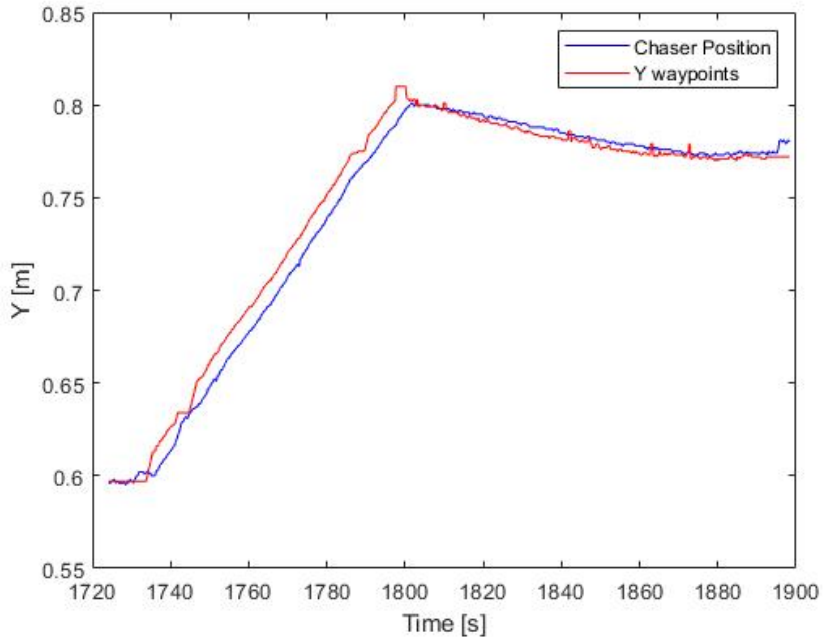


Figure 6.3 Y position data of capture of controlled target.

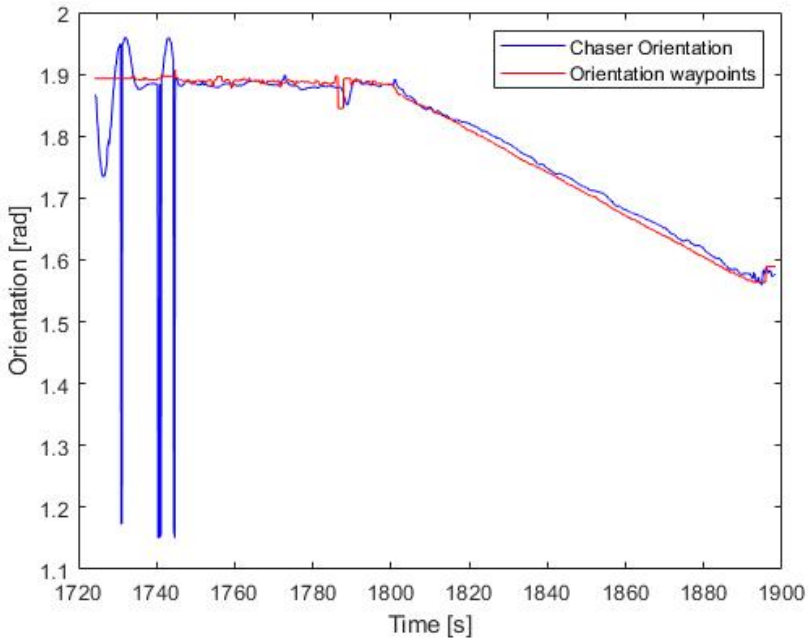


Figure 6.4 Orientation data of capture of controlled target.

The testing results are shown in Figure 6.1-Figure 6.4. Observing the results, we can see that the X and Y position follow the waypoints closely without being affected by the noise in the waypoints significantly. The robotic simulator lags the waypoints by the waypoint switching distance. In all the plots, the waypoints have noise due to the error in the camera therefore resulting in noisy way pointing. Observing the orientation plot, Figure 6.4, there are large deviations in the orientation of the chaser simulator at the start and end of the curve. At the start there is a significant swing in the orientation. This is due to the robotic manipulator re-setting to the start position. This applies a large torque on the chaser simulator due to the conservation of angular momentum. The chaser simulator can quickly correct for this disturbance. At the end there is a large variation in the chaser simulator's

orientation. This is caused by the contact and capture with the target simulator with the robotic manipulator. This causes the waypoint measurement to change drastically to correct for this disturbance applied on the chaser.

6.1.2 Stationary Uncontrolled Free-Floating Target

The second senecio is the stationary uncontrolled free-floating target. The thrusters and reaction wheel are used to control the target to prevent it from drifting but are turned off before the robotic manipulator contacts the target simulator. The drifting is caused by the imperfect level of granite table, which generates small gravity components in the surface plane of the table. In this case, the disturbances caused by the robotic manipulator contacting the target in addition to the drift are not compensated and can cause undesired motion of the target.

The chaser simulator was able to successfully capture the free-floating uncontrolled target. The simulator was able to successfully soft dock without instilling large unwanted forces onto the target simulator. Following the soft dock, the chaser was able to hard dock using the gripper fingers.

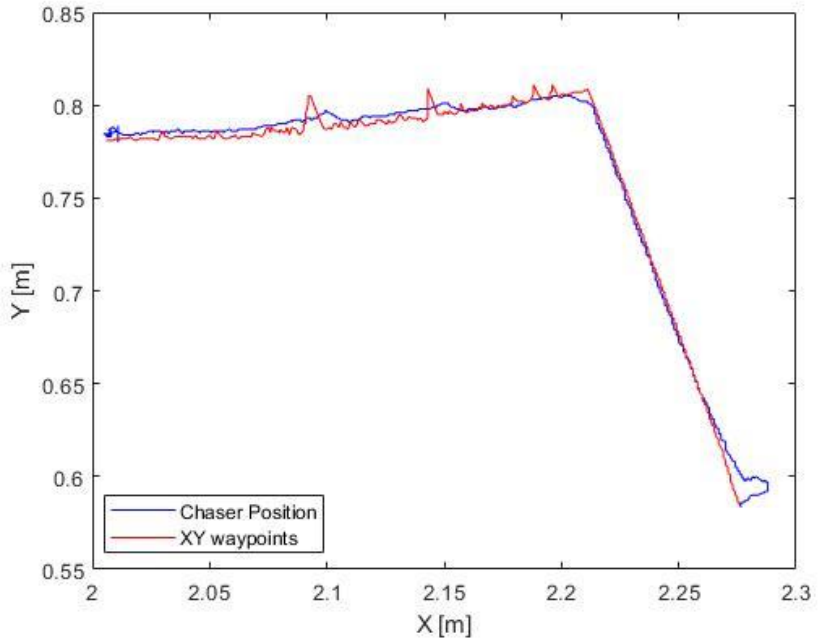


Figure 6.5 Position data of capture of uncontrolled free-floating target.

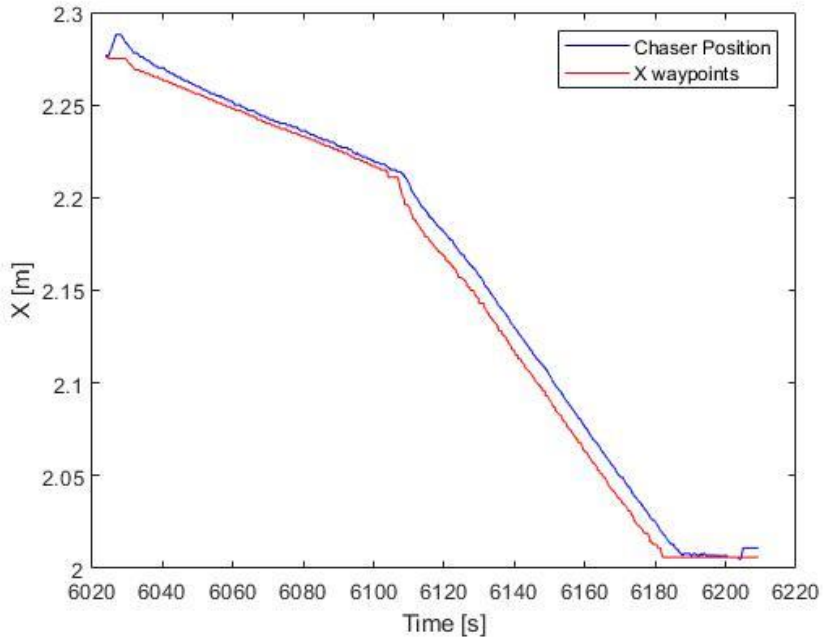


Figure 6.6 X position data of capture of uncontrolled free-floating target.

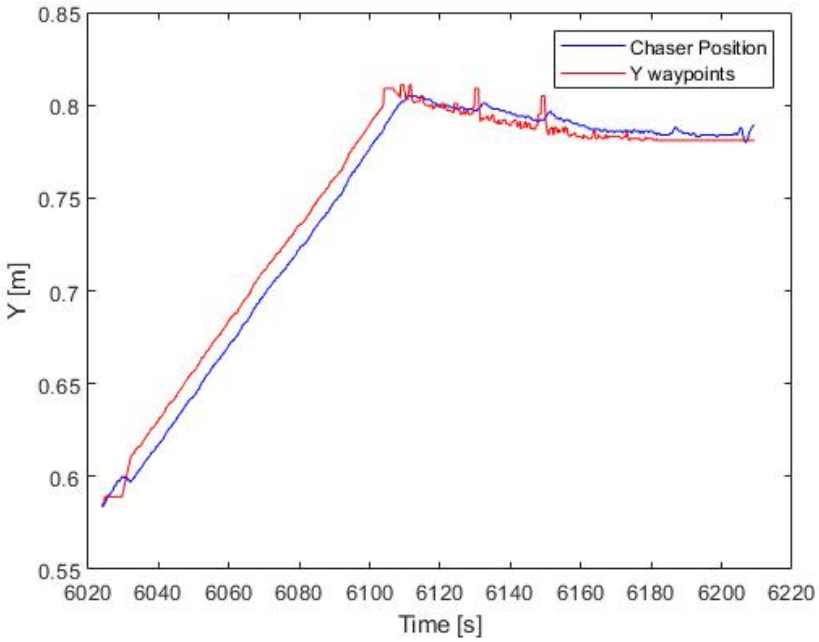


Figure 6.7 Y position data of capture of uncontrolled free-floating target.

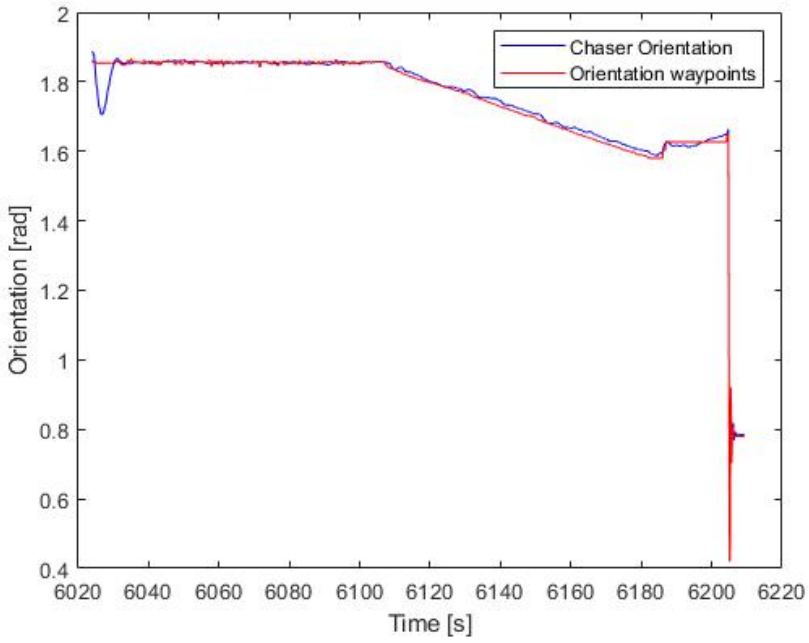


Figure 6.8 Orientation data of capture of uncontrolled free-floating target.

6.1.3 Controlled Free-Floating Tumbling Target

Finally, the capture of a controlled free-floating tumbling target is completed. This represents the worst-case scenario of a tumbling target where the target is tumbling, and its actuators will continue to provide torque to continue to rotate the target even after the target is captured.

The chaser simulator will synchronize its motion with the rotation of the target simulator and then perform the capture. After capture, the target will continue to tumble due to the reaction wheel maintaining its rotation speed. The chaser simulator will apply torques to detumble and stabilize the target simulator.

The chaser simulator was successfully able to approach and synchronize with the rotating target satellite. After capture, the chaser simulator successfully detumbled and stabilized the chaser/target pair, see Figure 6.9-Figure 6.16.

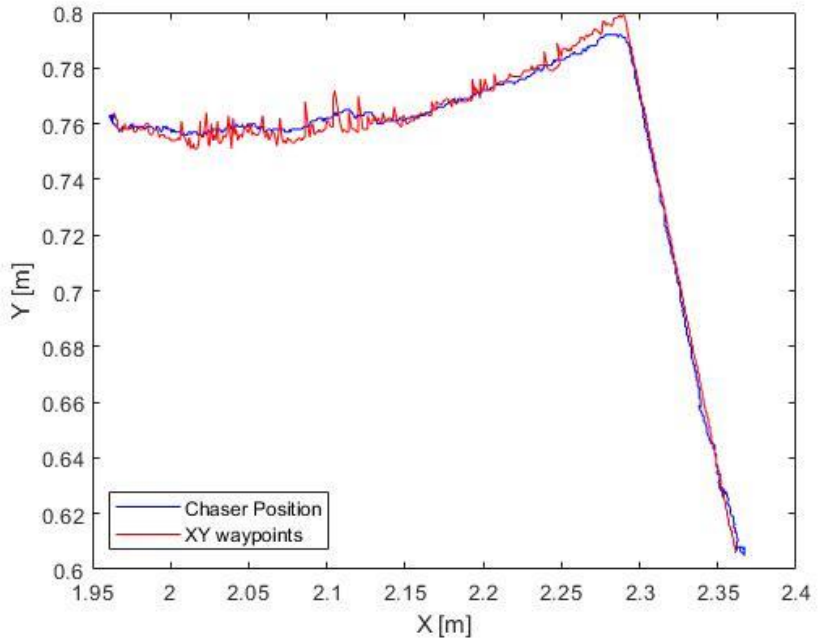


Figure 6.9 Position data of capture of free-floating tumbling target.

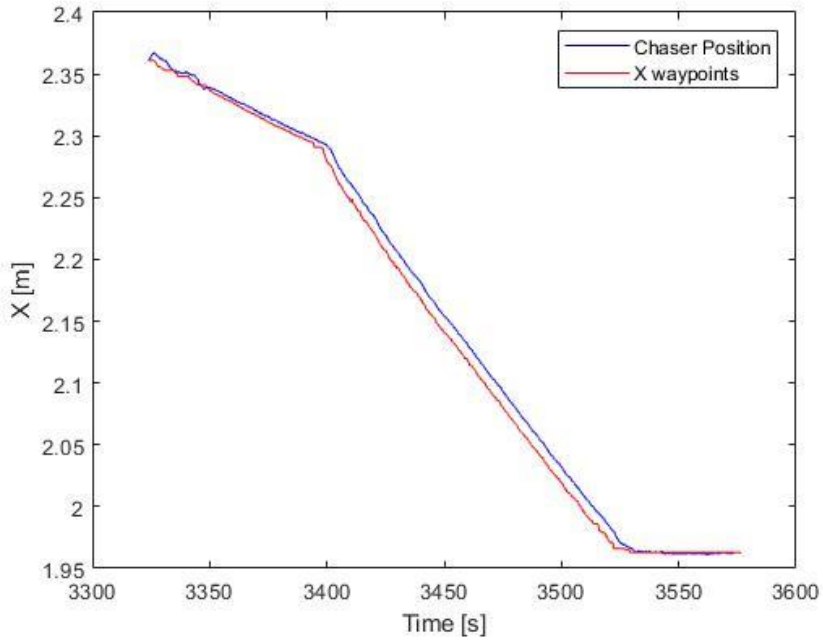


Figure 6.10 X position data of chaser capturing of free-floating tumbling target.

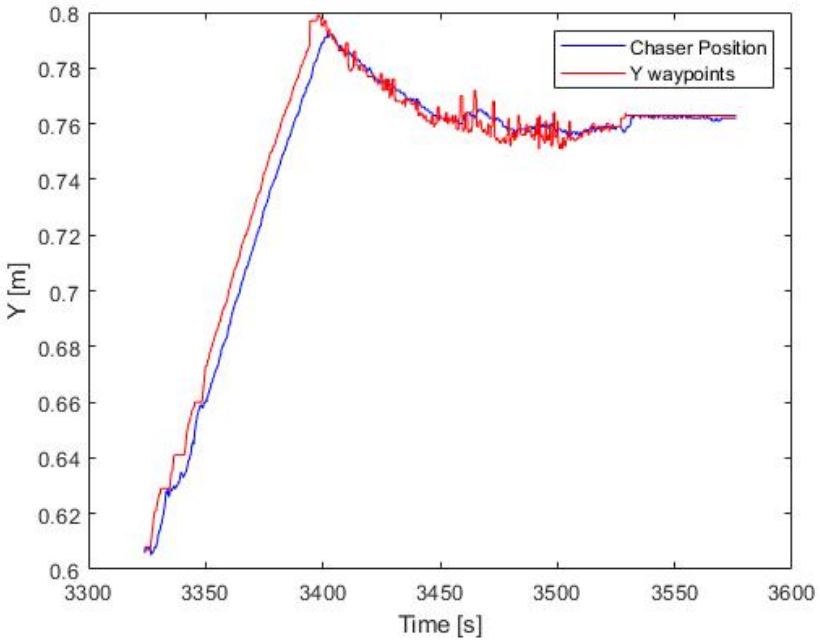


Figure 6.11 Y position data of chaser capturing of free-floating tumbling target.

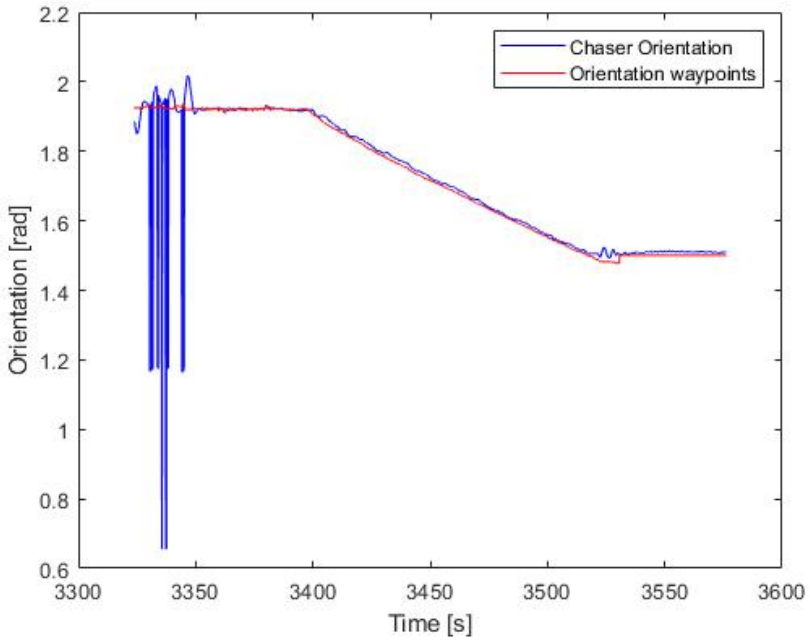


Figure 6.12 Orientation data of chaser capturing of controlled free flying.

The capture results for the rotating target simulator are captured below. The simulator was to rotate with an angular velocity of -0.04 deg/s. Computing the average of the recorded angular velocity results the measured average angular velocity during the uncontrolled phase was -0.03921 deg/s. The large spike in the position, orientation and angular velocity at approximately 1075 seconds is due to the chaser simulator capturing the target simulator, see Figure 6.16. Although the controlled target still attempts to rotate with the desired angular velocity, the detumbling sequence of the chaser manipulator hold the desired orientation of the target simulator. The average of the recorded angular velocity measured after the capture of the target simulator and detumbled was -0.00813 deg/s which falls within the noise of the sensor when it is stationary.

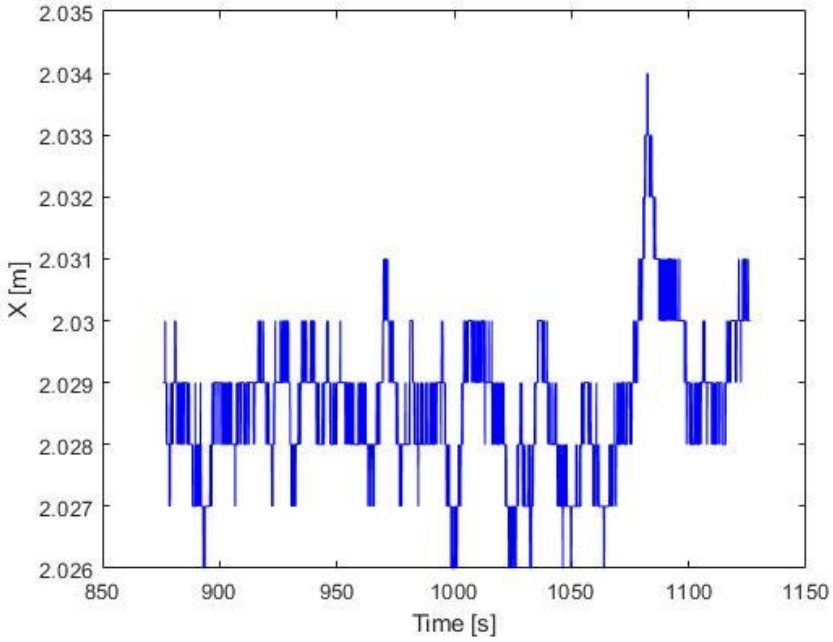


Figure 6.13 X position of the target simulator.

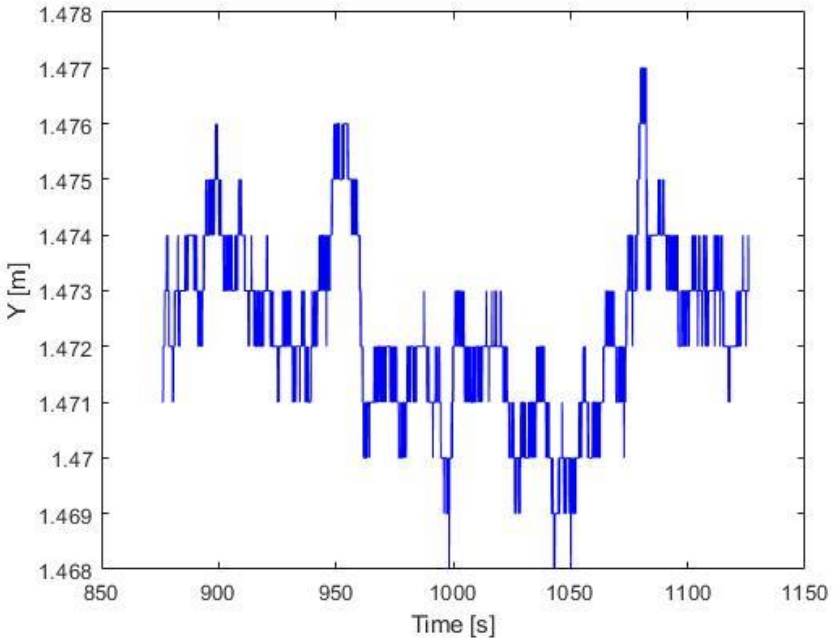


Figure 6.14 Y position of the target simulator.

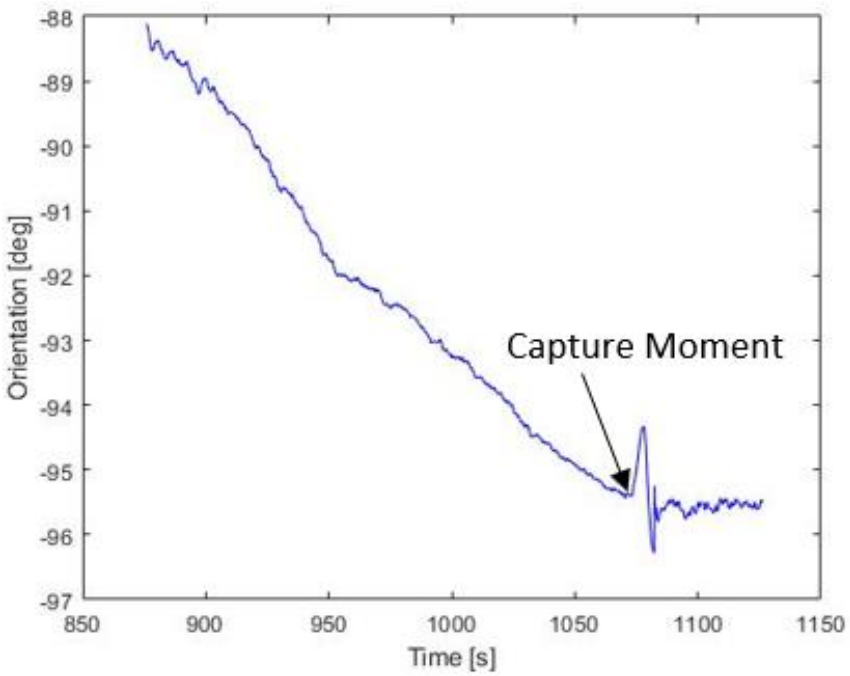


Figure 6.15 Orientation of the target simulator.

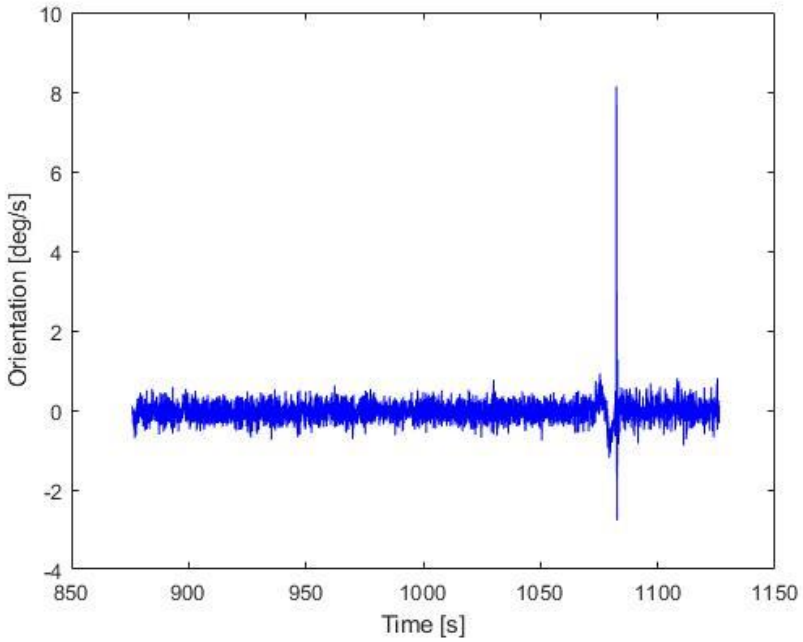


Figure 6.16 Angular velocity of the target simulator.

6.2 Improved Adaptive Controller Comparison

The adaptive controller using fuzzy gain scheduling was implemented on the satellite simulators to remedy some of the challenges with the coupling of the thrusters to perform both translational and rotational motion. The fuzzy logic is used to perform gain scheduling of the PD controller to activate thrusters when close to a waypoint when they would normally be inactive. To compare the performance between the two controllers the systems were only compared for the stationary capture case. This removes any additional variation in the results due to any errors arising from the rotating target simulator. In addition, for the rotating case, there were no changes to the way pointing and path planning

between the two versions. Only the control algorithm was changed showing that the system is modular. There is no improvement in the results expected for the circular path for the simulator as all three components of the chaser simulators pose will be changing on each loop iteration.

To compare the performance improvement, form the adaptive controller the total time to complete the linear and circular components is used. The adaptive controller will allow the simulator to follow the linear path more closely thus reducing the total length in the path the simulator follows and by extension the total time to capture. Both controllers were tested with the same initial conditions. The initial conditions were bought to within the tolerance of the systems way pointing. This adds additional variation to the results but are overall small. This was then repeated for different starting orientations and positions.

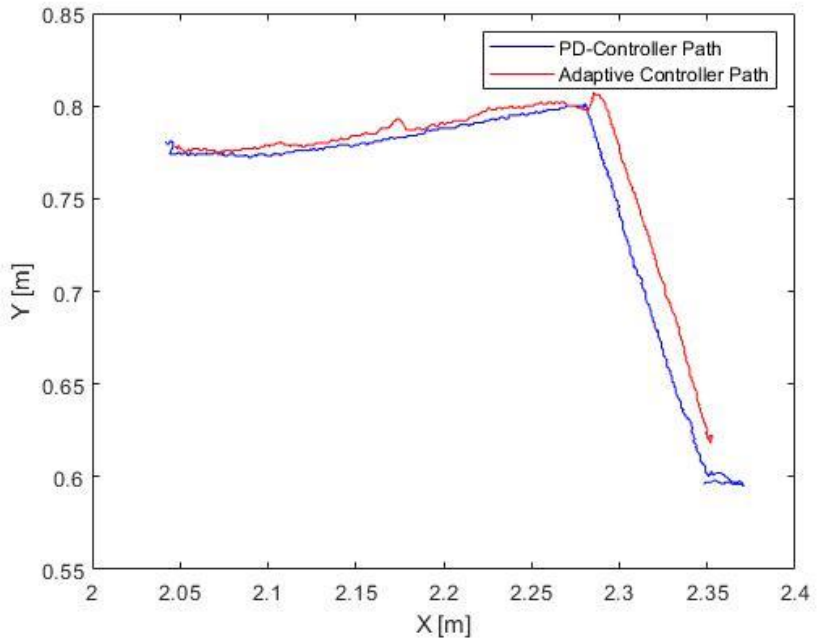


Figure 6.17 Run 1 PD and Adaptive Controller position comparison.

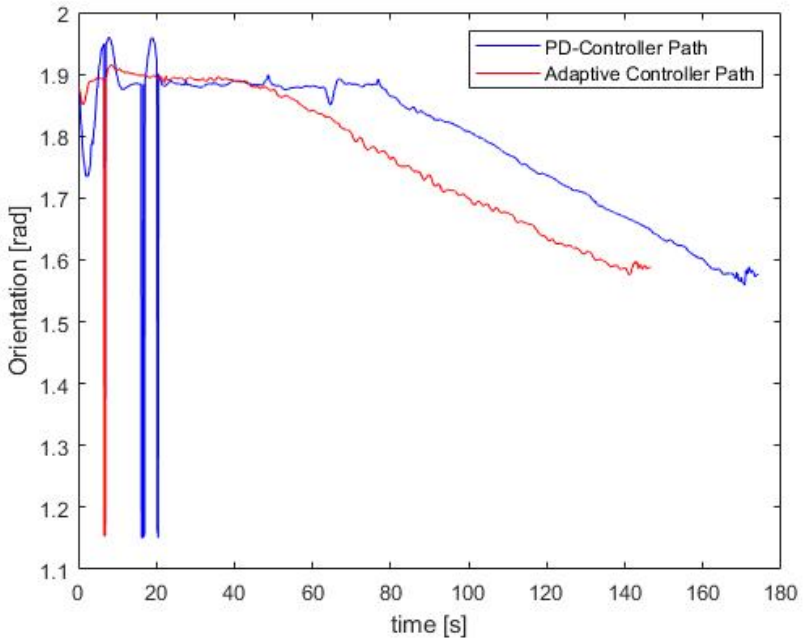


Figure 6.18 Run 1 PD and Adaptive Controller orientation comparison.

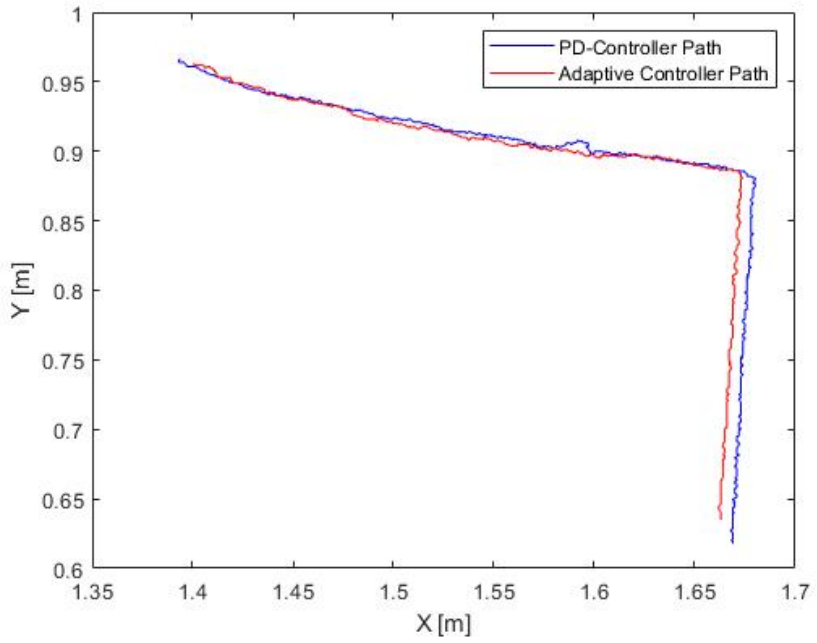


Figure 6.19 Run 2 PD and Adaptive Controller position comparison.

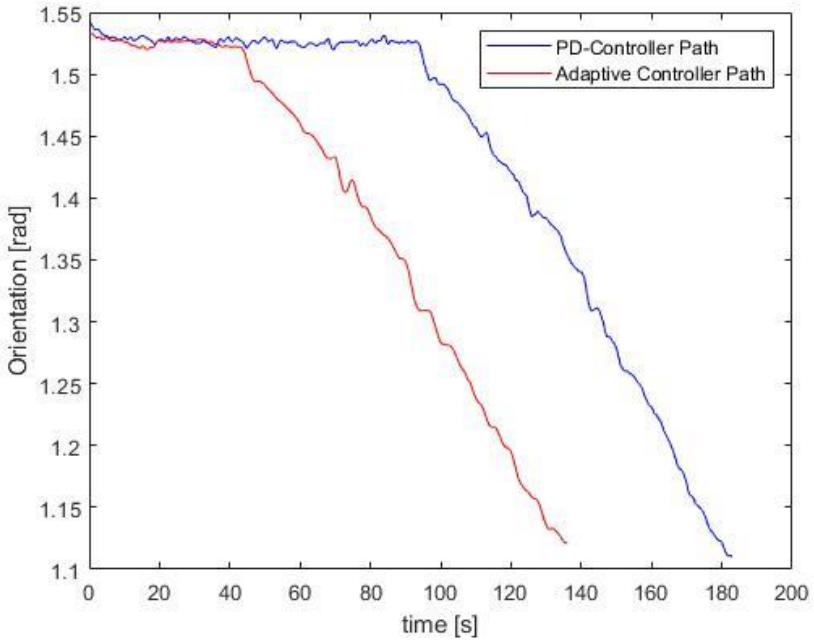


Figure 6.20 Run 2 PD and Adaptive Controller orientation comparison.

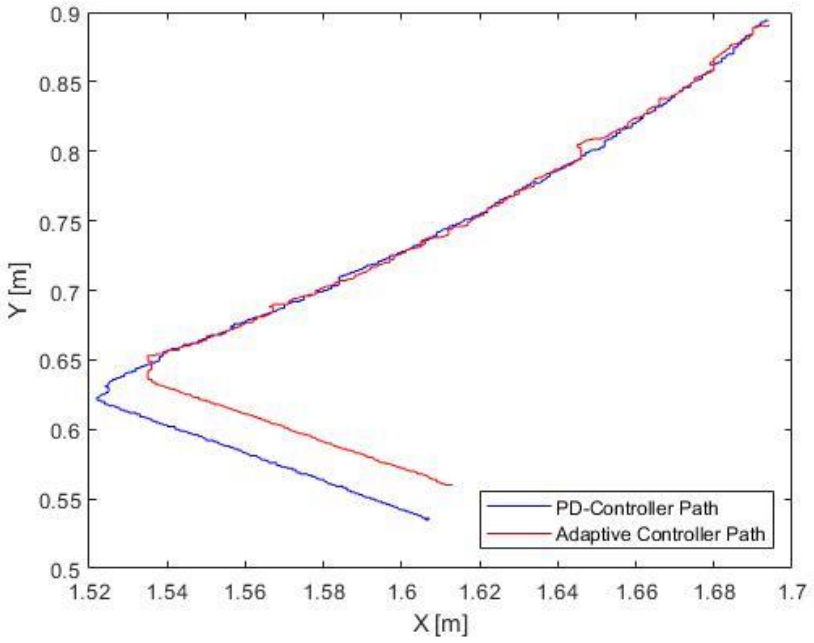


Figure 6.21 Run 3 PD and Adaptive Controller position comparison.

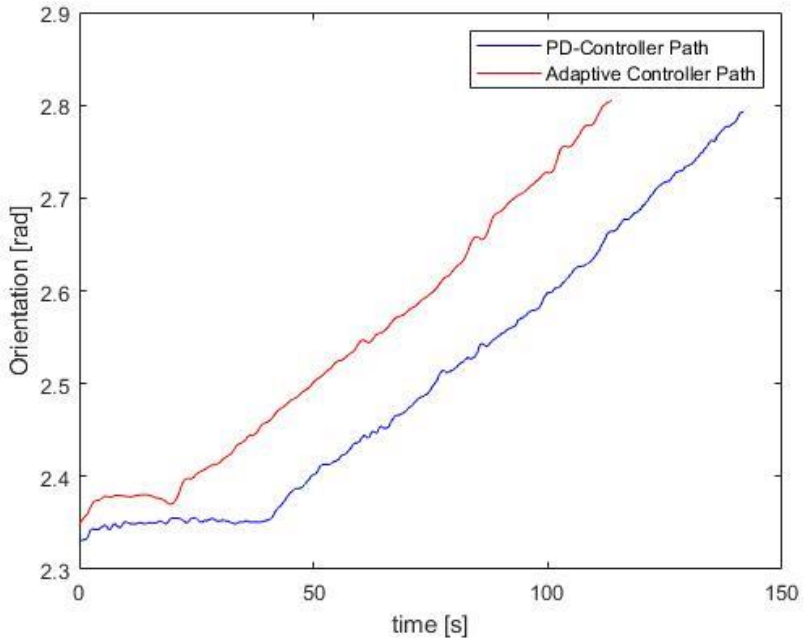


Figure 6.22 Run 3 PD and Adaptive Controller orientation comparison.

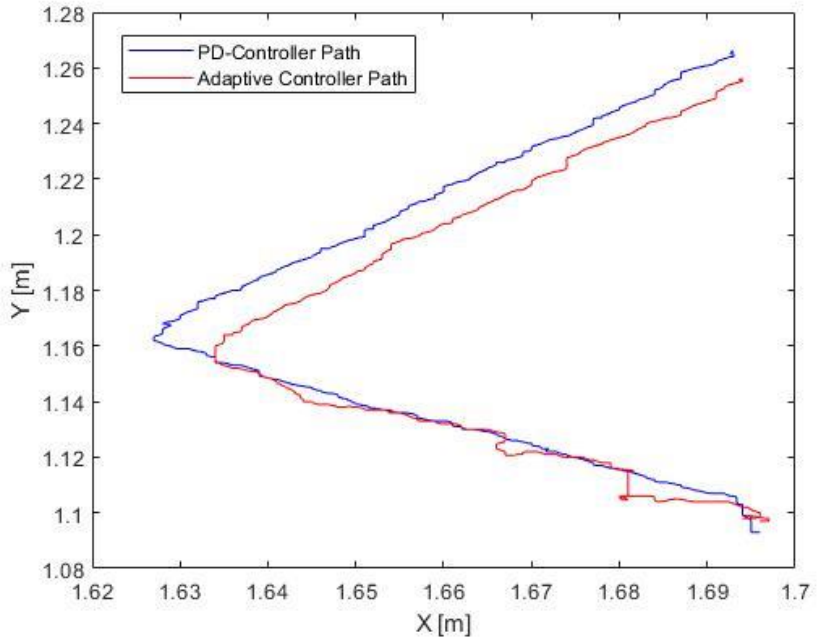


Figure 6.23 Run 4 PD and Adaptive Controller position comparison.

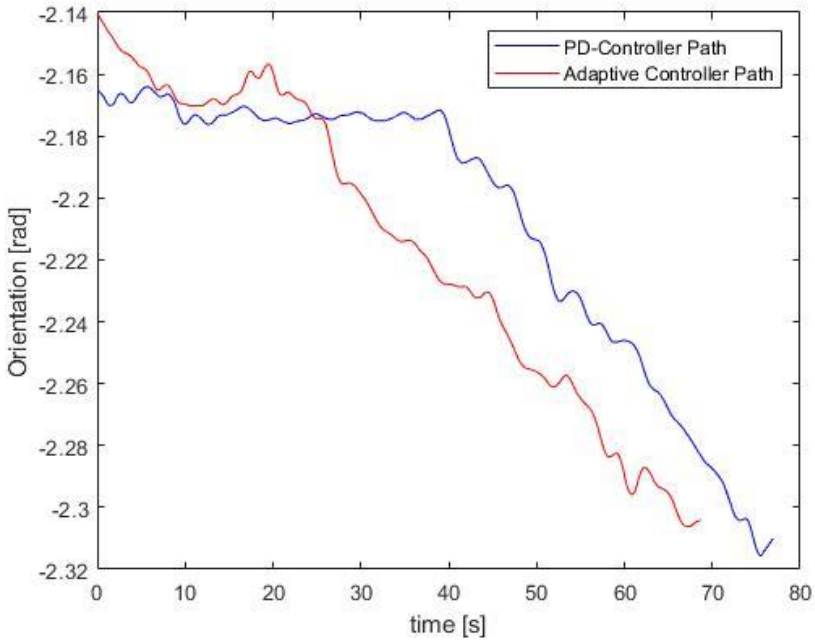


Figure 6.24 Run 4 PD and Adaptive Controller orientation comparison.

Taking the average percent improvements from the adaptive controller over the PD controller the performance improvements are determined. For the linear far range approach an average improvement of 47.1% was observed. For the synchronization maneuver, an average improvement of 0.63% was observed. The near zero improvement for the circular path and the significant improvement for the linear path was expected. Individual runs for the synchronization maneuver ranged above and below zero. This can be attributed to dust on the table slowing the simulator during individual runs.

Table 6.1 Adaptive and PD Result Comparison

Run Number	**Time**	**PD-Controller [s]**	**Adaptive Controller [s]**	**% Improvement**
1	Total	183.74	130.10	29.2%
	Far Range Approach	73.45	38.69	47.3%
	Synchronization Maneuver	68.01	62.89	7.5%
2	Total	176.78	130.68	26.1%
	Far Range Approach	89.23	42.25	52.7%
	Synchronization Maneuver	87.55	88.43	-1.0%
3	Total	138.35	109.37	20.9%
	Far Range Approach	34.772	18.22	47.6%
	Synchronization Maneuver	103.58	91.15	12.0%
4	Total	73.26	64.09	12.5%
	Far Range Approach	36.90	21.91	40.6%
	Synchronization Maneuver	36.36	42.18	-16.0%
	Average Total			22.2%
	Average Far Range Approach			47.1%
	Average Synchronization Maneuver			0.63%

The above results in this chapter show that all research objectives have been met. The testbed has been modified so that it is self-contained and can be used for OOS testing and development. Through the implementation of a target tracking vision system, PD controller, robotic manipulator, path planning and detumble algorithms the simulator was able to capture and detumble a target. Figure 6.1-Figure 6.16 show the capture of a stationary free-floating, controlled free-flying, and tumbling free-flying targets. The testbed is shown to be functional through these results. These tests verify the versatility of the testbed for a variety of capture scenarios.

To further demonstrate the versatility of the testbed an improved adaptive controller was implemented. The goal of the controller was to reduce the total time for the simulator to perform the straight-line path. The two controllers were tested for a variety of start conditions, see Figure 6.17-Figure 6.24. The adaptive controller was able to reduce the total time spent performing the straight-line approach by 41.7% on average. As expected, the controller had no improvement when performing the circular synchronization path. The adaptive controller had no significant computational load on the system and was able to reduce the time spent performing the straight-line manoeuvre. This shows that the testbed can be used to test a variety of control algorithms further showing the testbeds versatility.

It is important to note that many tests were attempted to compare the PD and adaptive controller. Due to the significant amount of dust on the table the simulator got stuck. This occurred for both the PD controller and adaptive controller. To obtain accurate results the table was thoroughly cleaned, and the ventilation system was turned off before each simulation run.

Chapter 7 CONCLUSIONS AND FUTURE WORK

Summary: This chapter consists of three sections. The first is a summary of the thesis accomplishments. The second section summarizes the overall results, contributions and impacts of the thesis work. The third section discusses the future work that will be investigated and legal considerations for on-orbit servicing technology.

7.1 General Conclusions

7.1.1 Thesis Accomplishments

This thesis developed several systems for an air-bearing testbed allowing for the testing of different on-orbit servicing technologies. Existing testbeds developed for the technology are heavily reliant on external computers to perform all positioning and tracking of both the chaser and target simulators. The information is then transmitted to the chaser and target simulators, where the chaser simulator then computes the paths and implements a controller. This is not a true reflection of real situation as the chaser satellite would compute its own localization, and track and compute the target satellites position,

orientation and spin rates. The thesis work outlines the upgrading of an air-bearing simulator for OOS testing that is truly autonomous. The air-bearing system was equipped with an attached robotic manipulator to perform the far-range approach, capture, and de-tumbling. A side-facing monocular camera was developed to perform the tracking of a target to determine its position, orientation and angular velocity.

7.1.1.1 Robotic Simulator Upgrades

The original robotic simulators were developed by Tsinghua University originally [85] and was further developed by visiting professor Ning Chen, Sat Li for his PhD and Joshua Cookson for his master's degree [82]. Further repair and upgrading were required. The on-board computer was upgraded to handle the additional image processing to be computed. In addition, several thrusters, valves and tubing were damaged and leaking and thus were replaced.

7.1.1.2 Gripper Development

A gripper was developed consisting of two fingers with a triangular shaped structure. The triangular structure guides the manipulator to the target where it can soft dock with the target by making contact in the cone like region. Then gripper fingers can then reach around the capture point and hard dock with the targets capture point making a secure connection.

7.1.1.3 Robotic Manipulator Development

To move the gripper towards the target capture, point a robotic manipulator was developed. The length of the manipulator and mass was developed to fall within the criteria of past and planned missions. The forward and inverse kinematics of the manipulator was

computed. A path planning algorithm was created for the manipulator to tend toward the target waypoints. Using an Arduino Mega micro-controller, the servo motors needed to be programmed and controlled. A customer PWM program was required to actuate the servo motors. To power the robotic arm a step-down voltage converter was added.

7.1.1.4 Star Tracker Correction

The star tracking system was developed by the previous master's student Josh Cookson. The star tracking system unfortunately suffered from errors in the computation of its position. A checking program was developed to determine if the new computed position and orientation was an error.

7.1.1.5 Tracking Camera Development

To track the target satellite simulator, a camera is placed on the side of the chaser satellite simulator. The tracking module was written in C++ using OpenCV and the Aruco Library. An Aruco marker placed on the target satellite. To send the camera information to the main LabView code, the Cesanta Mongoose Embedded Web Server [89] is used and an XML file containing the cameras tracked position and orientation measurements using an HTTP/GET request.

7.1.1.6 Path Planning Software Development

To determine how the simulator will approach and synchronize with the target, a path planning procedure must be determined. To keep the path computationally simple, the closest straight-line path is used to make the far-range approach to the target. Then a circular synchronization maneuver is used for the face with the robotic manipulator to align with the capture face of the target simulator. The straight-line path and the circular path

represent the best case and worst-case path scenarios that the controllers would need to follow.

7.1.1.7 Controller Development

A PD-controller was initially designed to follow the path waypoints outlined by the path planning algorithm. The controller stability criteria and gains were computer. Then a Matlab & Simulink simulation was created to test all algorithms and further adjust the gains.

Following the PD-controller, an adaptive controller was developed to correct the issues encountered by the coupling of the thrusters and the PD-controller. Fuzzy logic was used to adjust the gains that were fed into the PD-controller. The controller showed improvement when doing the linear far-range approach. This demonstrated the modularity of the satellite simulator system and its ability to interchange algorithms and components to test different aspects of on-orbit servicing.

7.2 Contributions of Thesis Work

This thesis has developed an on-orbit servicing testbed that improves on existing testbeds. This work was motivated by the need to have a fully autonomous testbed to perform ground testing and verification of on-orbit servicing algorithms and hardware. Existing testbeds are not truly autonomous as they are reliant on external sensors and computers to determine the position and orientation of both the chaser and target satellite simulators. These external computers then transmit the target satellite simulators position and orientation to the chaser which then performs path planning and control onboard. With

the external positioning systems, the position and orientation of both the target and chaser are updated at a rate of 50-1000 Hz with very high accuracy and precision. This does not represent a the real on-orbit servicing scenario. First, all position and orientation information are computed on the on-board computer of the chaser satellite. This includes a sensor to determine the target simulators position and orientation onboard the chaser satellite. This imposes high computational loads on the chaser satellites on-board computer that is not considered with all existing testbeds.

The newly developed testbed is fully autonomous using a star tracking system to determine the chaser simulators position and a monocular side facing camera to determine the target simulators position and orientation all computed on-board chaser simulator. The simulator was designed to be modular in both its hardware and software. Individual components of its software were developed to be replaced with different algorithms. An example of this was performed where the PD-controller was replaced with an adaptive controller showing and improvement in the simulators ability to perform the far-range approach towards the target simulator.

The robotic manipulator is also fully modular where the individual lengths and gripper can be easily extended or interchanged with different designs. This allows for different lengths of manipulators to be tested in the future, with redundant degrees of freedom, and gripper designs.

This test platform can used to test various path planning, control, vision, and detumble algorithms for a free-flying chaser for OOS. The testbed can test the capture of

three target cases; un-controlled free-floating target, controlled free-flying target, and tumbling target.

7.2.1 Testbed Limitations

7.2.1.1 Table Limitations

This testbed has several limitations that must be considered when performing future experiments. The first limitation is common to all air-bearing simulators with robotic manipulators as it is only able to simulate in 3-DOF. OOS vehicles operate in 6-DOF making path-planning and control algorithms more complex.

The current table is settled to 0.2 deg inclination of the top surface of the table. This results in an acceleration of 3.08mm/s^2 in the plane. The table level cannot be easily corrected as one of the legs is damaged. In addition, the lab does not own a level with high enough accuracy to appropriately level the table. The thrusters of the simulator can overcome this acceleration therefore leveling was not required. As the testbed is not in a cleanroom environment, dust builds up on the testbeds surface. The dust causes additional disturbances on the simulators. The combined level and dust issues makes testing of the free-floating chaser simulator impossible.

7.2.1.2 Thruster Limitations

Currently, the thrusters are limited to on-off activation on each loop iteration. To improve the thruster actuation and provide better control the thrusters can be activated using PWM. Using PWM allows for the approximation of variable force output as a

function of the PWM duty cycle. Optimization can be performed to maximize force output and conserve fuel [78].

A further improvement to using PWM with on-off thrusters would be to replace the on-off thrusters with a variable/proportional solenoid valve. The thrusters used in the testbed do not mimic operational thrusters used in orbit. The general thrust profile for gas-get thrusters has a lag, rise and fall time for the build up of propellant flow, see Figure 7.1.

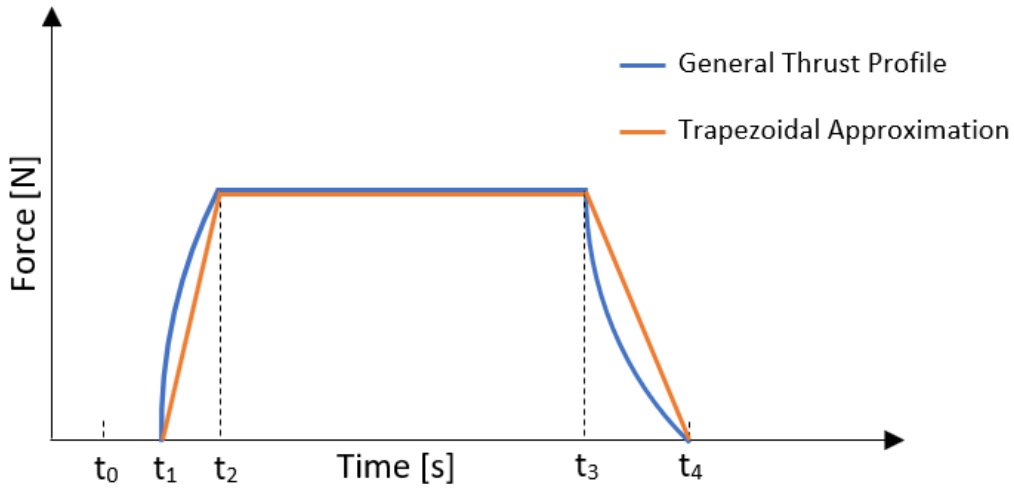


Figure 7.1 General thrust profile for gas jet control systems [100].

The lag of the thruster can be easily simulated by providing a delay in the thruster activation. If a variable solenoid valve or a proportional solenoid valve, it can approximate the thrust profile, see Figure 7.2. Using the trapezoidal approximation, the output pressure can be increase in steps until the desired force thrust force is achieved. A variable solenoid valve has a limited number of steps. A proportional solenoid valve adjusts its pressure

using a PWM input, providing a smoother thrust profile which can simulate the general thrust profile more closely.

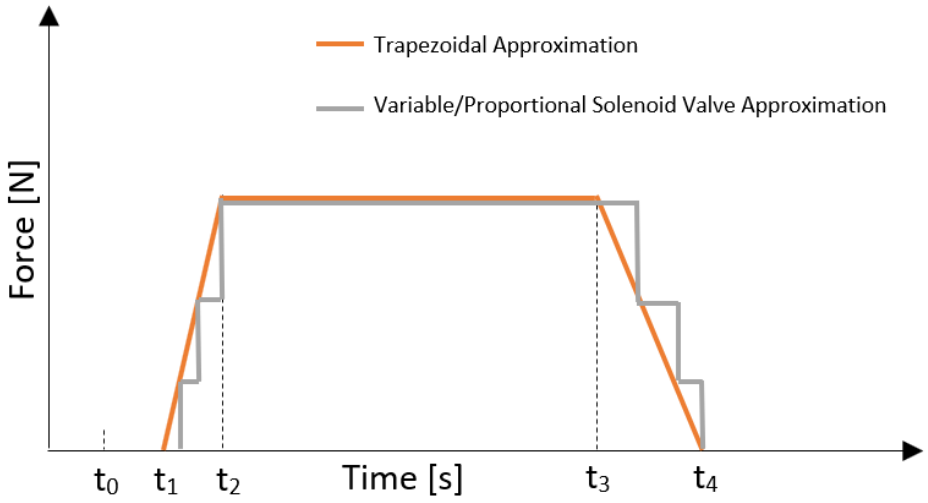


Figure 7.2 Variable/Proportional solenoid valve thruster approximation.

7.2.1.3 Tracking Camera Limitations

Another limitation is with the Aruco marker. Having a specified marker on a spacecraft is highly idealized as the majority of spacecraft launched have no standardized target features. This system is only capable of detecting Aruco markers. In the future, a vision system using artificial intelligence (AI) is required to detect any space object and determine its position and orientation. An additional limitation in using the Aruco marker is its ability to accurately detect the target. Depending on the size, location, orientation or the target and camera used the accuracy of the measured position and orientation results will decrease. As the number of target markers increase, the computational load on the system also increase. This can limit the number of targets that the system can view at a

given point in time. Below are sample results to demonstrate how the Aruco marker results would vary under different conditions.

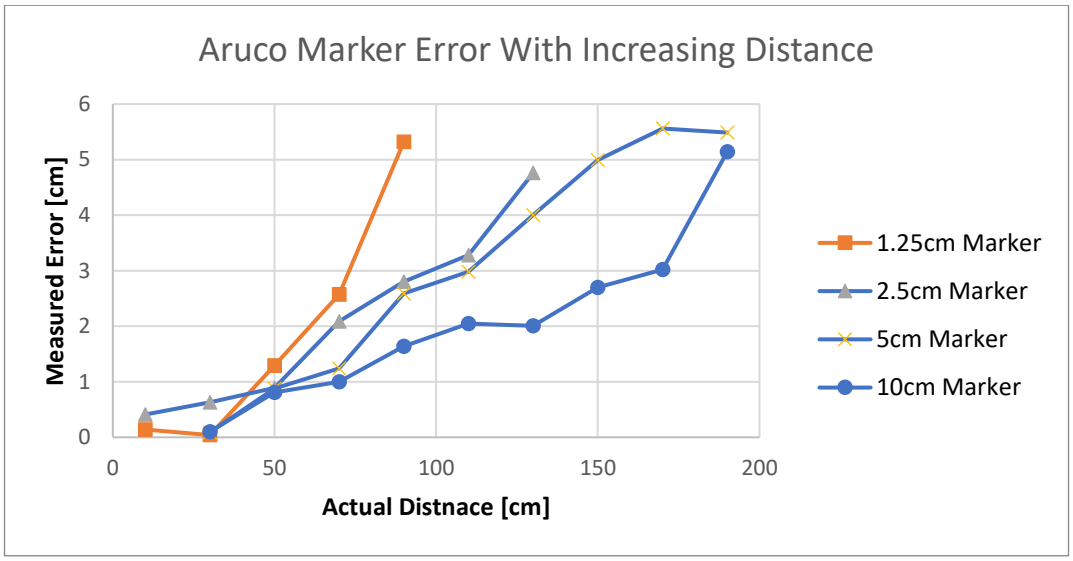


Figure 7.3 Aruco marker error with increased distance and size of marker.

Figure 7.3 shows the increased error of the Aruco markers determination of distance away from the camera with various sized markers. As the marker increases in size, at a given distance the error is reduced. Here the 10 cm marker had the lowest error. However, due to the size limitation of the simulator only a 5 cm marker can be placed. It can also be seen that the larger markers, when placed close to the camera were too large and were unable to be detected. This was the case with markers greater than and including 5 cm. The simulator would never approach this distance as it would cause interference with the robotic manipulator and the target simulator before this point.

Another issue with the Aruco library is when the marker is out of the camera frame. The library performs an extensive search for the target markers causing a significant drop

in the update frequency of the tracking system. This is shown in Figure 7.4 when the update frequency drops below 5 Hz.

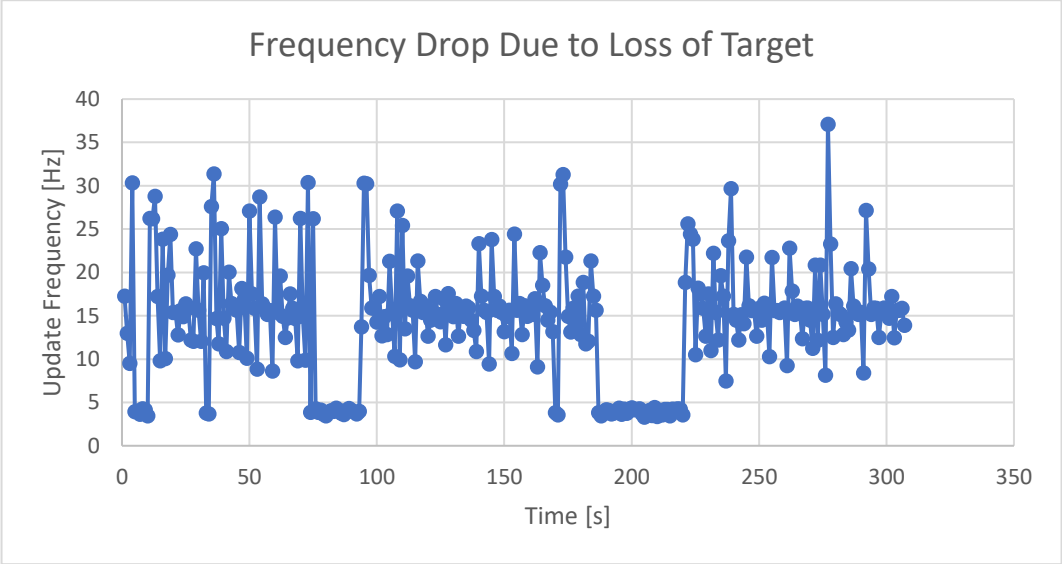


Figure 7.4 Drop in Update from tracking camera.

7.3 Future Work

From the course of this thesis study, the following areas are considered worthy of further studies.

7.3.1 Free-Floating Chaser at Capture

As mentioned in section 1.3 Limitations of Existing Research Methods, an important consideration during the capture phase is the attitude control system (ACS) should be turned off [9]. The reason for this is the contact forces during the capture phase can be considered random and would cause the random triggering of the ACS system. This

could lead to shock and damage of the spacecraft, manipulator or target satellite [27,58]. This operation mode also leads to fuel savings and therefore extends the overall duration of the mission. This area is actively investigated and methods of using the redundant joints of the robotic manipulator to cause the manipulator to tend toward its target using the principle of conservation of momentum. This method must overcome the challenge of having both kinematic and dynamic singularities.

7.3.2 Target Identification and Tracking without a Target

One of the most actively investigated and most important technologies that are under investigation for on-orbit servicing is identification of a target, determine its position and orientation and the identification of a capture point. The goal is to achieve these without any specified target features. The reason for this is that there are many active satellites and debris that require capture but do not have these specified features. For new spacecraft launched, there are no standardized features that are implemented. The European Space Agency is attempting to standardize target features for all new spacecraft launched but to effectively use on-orbit servicing to its outmost performance, the servicer spacecraft must be able to capture all possible types of spacecraft and debris.

There are many approaches being attempted primarily focusing on using lidar and serio cameras combined with machine learning to perform the object identification, determination of its orientation, spin rates, and optimal captures points.

7.3.3 Multi-Arm Manipulators

Multi-arm manipulators are more practical for most on-orbit applications as the multiple arms allow for more dextrous and complex operations [94]. Some other uses of multi-arm manipulators are the LEAP experiment and for attitude stabilization [90].

The Stanford University team recognized that the use of gas thrusters should be minimized to conserve fuel, so they propose the Locomotion Enhancement via Arm Push-Off (LEAP) where the robot manipulators would push-off of objects in space to throw itself to a new location. Using cyclic motions with the arms the spacecraft can be re-oriented in space to grasp other objects [90].

The additional arm can also be used to compensate the attitude of the base spacecraft. When one arm is performing a maneuver to grasp an object, the second manipulator can be used to directly counteract the first manipulator resulting in a zero-net rotation about the spacecrafts center of mass [10,101].

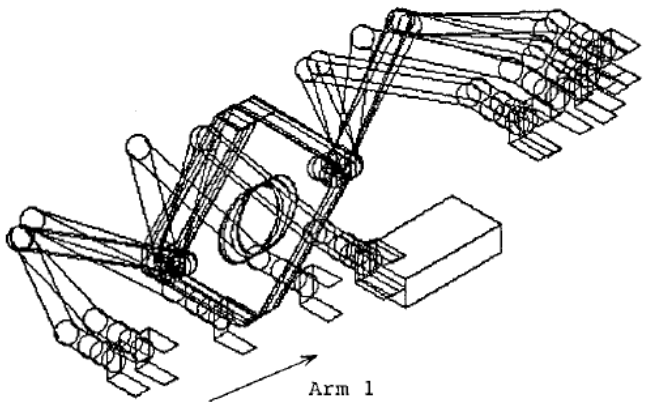


Figure 7.5 Coordination of dual arms to reduce net torque on the system [101].

A challenge in the control of multi-arm manipulators is when the manipulators grasp a common object creating a closed loop. In this case, the set of dynamics used for the open-loop manipulators needs to be adjusted for the closed-loop scenario. The operational space for the closed-loop system is reduced [102] as it operates as a single parallel manipulator.

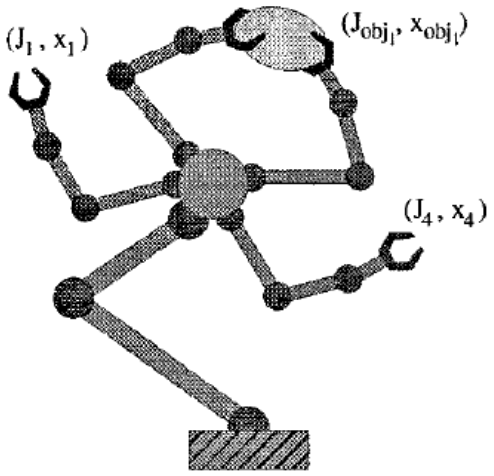


Figure 7.6 Serial-to-Parallel manipulator grasping a common object [102].

7.3.4 Flexible Manipulators

With space manipulators becoming incredibly lightweight, we can no-longer consider the links as rigid bodies. Each link and joint have some level of flexibility which can limit the manipulator in its motion [10]. Due to this limitation, the study of flexible manipulators is an important area of study requiring more research.

In addition, the study on how these flexible manipulators play a role in assembling of space structures is an important area of study. These spacecrafts would act as connecting

joints for large structures and would have vibrations throughout the space structure [10]. In the case of multi-arm robots, the total system with the second arm makes the system flexible [28].

In the paper *Parametric Analysis of a Controlled Deployable Space manipulator Used For Capturing a Non-Cooperative Satellite* the manipulator is modeled with a spring dampener on the last link of the arm, with an impedance PD controller. The PD controller allows for the control of the free motion and the impedance control compensates for the external environment. The end-effector acts like a mass-spring-damper system. Through simulation, the authors found that the targets contact forces and velocities fell to sufficiently small values after 15 seconds. It is worth noting that the authors conducted a second simulation where the solar panels were simulated as flexible structures and found that their vibration does not have a large impact during the contact phase. The contact forces and velocities fell to sufficiently small values after 20 seconds [92].

7.3.5 Capture of a Non-Cooperative Tumbling Target

A non-cooperative tumbling target is one that is spinning in an undesirable manor for capture. This poses the additional challenge in the planning and capture phase of the servicing spacecraft. According to JAXA, if a target is tumbling at a rate below 3^{o}/s, it can be captured easily. Between 3^{o}/s and 30^{o}/s, the target is considered tumbling and can be de-tumbled. Beyond 30^{o}/s the spacecraft should not be considered as a target [93]. The additional step of de-tumbling the target spacecraft is required.

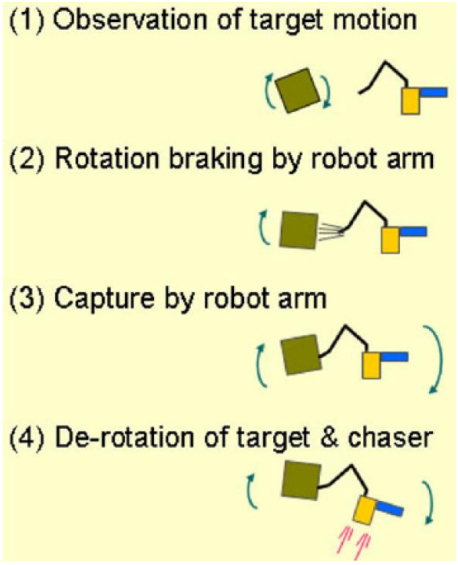


Figure 7.7 Typical target capture scenario with de-tumbling [93].

Most methods of capturing a tumbling target require the chaser to be free-flying as the ACS is used to synchronize the attitude of the chaser to the target spacecraft, which as stated before using ACS is undesirable. In addition, the ACS of the chaser spacecraft used to minimize the angular momentum transferred to the chaser spacecraft. Additional work is required by the ACS to cancel the total momentum of the combined chaser target system after capture. Not only does this require large amounts of propellant to complete the maneuvers, for higher angular velocities the ACS of the chaser may not be able to compensate for the total momentum of the combined chaser and target system [27].

In the paper *Trajectory Generation Method For Robotic Free-Floating Capture of a Non-Cooperative, Tumbling Target* [27] a method where the motion of the manipulator is optimized to cancel out the momentum of the system during the approach phase. The methodology is to pre-load angular momentum using the ACS on the chaser that is opposite

to that of the target. Then the momentum is redistributing to the manipulator prior to contact. The amount of momentum transferred to the manipulator depends on the capture strategy. At capture the total momentum of the system becomes zero [27].

7.3.6 Multi-Robot System (Swarms)

A popular area of robotics is in multi-agent swarm robotics. A swarm consists of a team of robots that share tasks and coordinate to achieve a goal. Swarm technology can also be implemented in space free-flying robotics for on-orbit servicing.

Instead of having one large multi-tooled SFFR, multiple smaller SFFR's can be deployed which saves weight, saves space and provides redundancy amongst the swarm increasing the likelihood of mission success. Reducing weight is important since with it is directly proportional to launch cost [103].

A swarm can be used to assemble large space structures cooperatively or perform related tasks in repairing satellites. SFFR's in a swarm can also be used for asteroid mining to collect raw materials [104]. Although swarm technology is well studied for Earth based tasks, they do not consider the dynamic and physical interactions between the team of robotics and the elements they will be manipulating. In the paper *Coordinated Control of Space Robot Teams For The On-Orbit Construction of Large Flexible Space Structures* [105], the authors propose a control and path planning methodology to address this issue.

7.3.7 Capture Mechanism Design

One of the largest challenges with SFFR for repair and de-orbiting missions is the fact that satellites are not designed with dedicated gripping features. Different satellites

would also have different grasping features making it impossible for a robotic gripper to capture all satellites.

For future space missions, the European Space Agency proposed the Clean Space Initiative in 2012, where a passive mechanical interface for capture should be developed [1] that can be placed on future spacecraft for a common capture interface. There are two main groups for capture mechanism. The first is the probe and drogue mating system. The conical feature of the target spacecraft is used to guide the probe toward a socket. In Figure 7.8, the petals are actuated to soft capture with the target and then the hooks are used to hard capture the target.

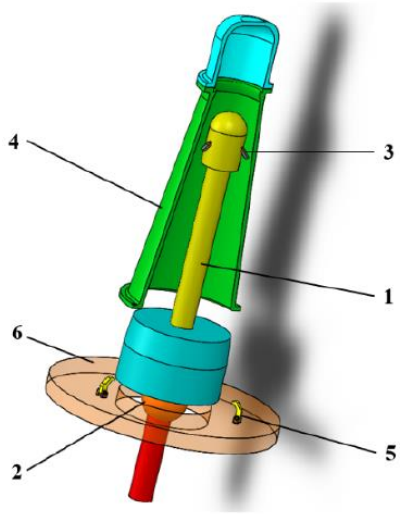


Figure 7.8 STRONG mechanism for SFFR capture. Probe (1), Universal joint (2), petals (3), passive interface (4), hooks (5), and flange (6) [1].

The second method is using a finger like grasping system to capture a target. This mechanism is traditionally used on the historic SFFR missions listed in Table 2.1.

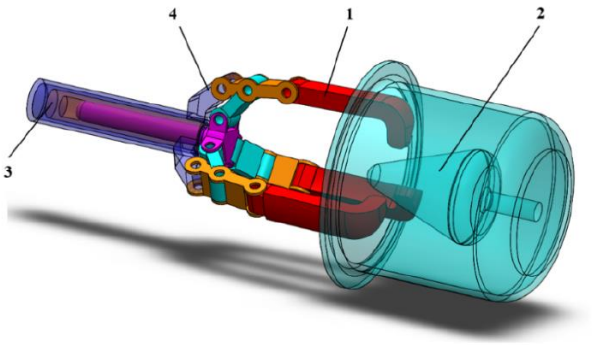


Figure 7.9 Finger mechanism concept. Fingers (1), passive interface (2), linear actuator (3), and fixed external frame (4) [1].

7.3.8 Legal and Regulation Building

Between the years 1965 to 2010, an average of 118 objects per year were launched into space [106]. This placed 5385 satellites into orbit as of 2018. In 2018, there was a total of 242 non-anomalous on-orbit breakups, increased from the 201 recorded in 2007 [107] and 194 recorded from 2004 [108]. This number will continue to increase rapidly due to the rapid growth of the space industry for private and defense activities. In 2019, India tested a missile defence system that shot down a low earth orbiting satellite which produced 60 trackable pieces of debris. Of the trackable pieces of debris, 24 of them pose significant risk to the International Space Station [109]. Activities such as theses tests conducted by India will rapidly increase the number of debris and the risk of collisions.

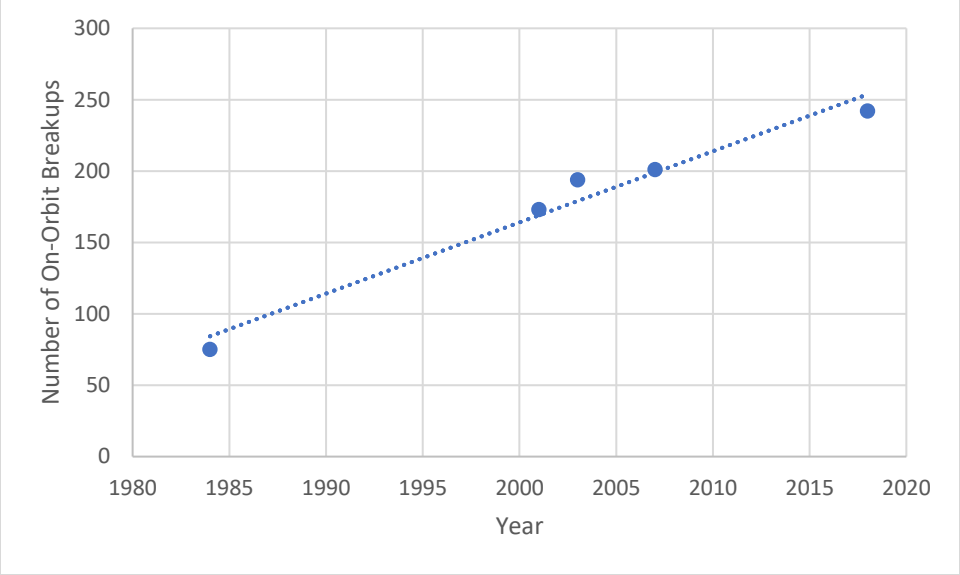


Figure 7.10: Number of non-anomalous on-orbit breakups [107,108,110].

There are many legal, regulatory and political challenges with the implementation of on-orbit servicing and active debris removal spacecraft. No private company can perform OSS or ADR work without licensing and supervision from the government which the company resides and of that of the space object. When a private entity performs OSS or ADR, they are acting as representatives of their government, and therefore their government are responsible and liable for their activities [111,112].

The paper *Regulatory Framework and Organization For Space Debris Removal and On-Orbit Servicing of Satellites* [111], outlines these issues in detail and is briefly summarized here. When a private company launches a space object, it must register the object to its government registry as well with the United Nations. Ownership of such object belongs to said State regardless of the object's functionality. Permission to perform ADR

and OOS on the space object in question is required unless that State has abandoned the space object. The State remains liable for damages caused by the object. The object is subject to salvage rules of international maritime law.

Regulations imposed by other states must be considered when planning ADR and OOS activities. For example, the space object may be subject to the U.S. International Traffic in Arms Regulations (ITAR). This requires approval by the U.S. government to acquire technology and data before being transferred to a foreign person. For OOS this is an important consideration as information of technology and data would be required to perform repair tasks. For ADR transferring of control of a satellite (or its components) and its disposal require prior approval. For ADR, if the object re-entries and causes damage, injury or death on the surface of the Earth it is the liability of the launching State [111] which could deter a State from having their satellite de-orbited. International regulations and procedures are required to create standardized practices and procedures in on-orbit activates instead of discouraging states from having their satellites de-orbited, encourage them to clean up the space environment.

7.3.9 Entrepreneurship Activates

Robotic spacecraft can perform many of the tasks that would be considered too dangerous or costly for an astronaut to perform. Unlike astronauts, robots do not require life-support systems and can operate for longer durations. The use of robotic spacecraft also alleviates the risk of death in the event of an emergency from an astronaut performing and extravehicular activity.

On-orbit servicing robotics has been investigated since the early 80's [9], but research slowed due to financial viability. With advancement of technology and the large communication constellations planned, the implementation of the technology is financially viable and cost effective for the customer. To reduce life-cycle costs, a satellite could be built with lower reliability and subscribe to a robotic servicing [9,21,113]. Satellites maintained by servicing robots can extend their operational life by as much as 15 years [49].

Studies for geostationary satellite repair suggest that satellites must become smaller, lighter and cheaper for OOS to become cost-effective. The cost associated with the reduced reliability, including the subscription to OOS is smaller than the savings in manufacturing and launch. The studies suggest that re-fueling missions will always be financially viable but repair missions for life extensions will not always be financially viable [114]. These studies focus on geostationary satellites, and do not consider the growing space of LEO communication satellites, which are becoming the fastest growing sector in space.

In 2013, there was a sudden increase in objects launched largely due to the increased uses of CubeSats. Particularly, in the past two years (2017 and 2018) a dramatic increase of 453 and 382 satellites were launched respectfully. The number of space objects will continue to increase significantly in the upcoming years. Many companies are currently in the development and implementation of large-scale constellations for telecommunications and Earth observation. Thousands of satellites will be added into the Low Earth Orbit environment.

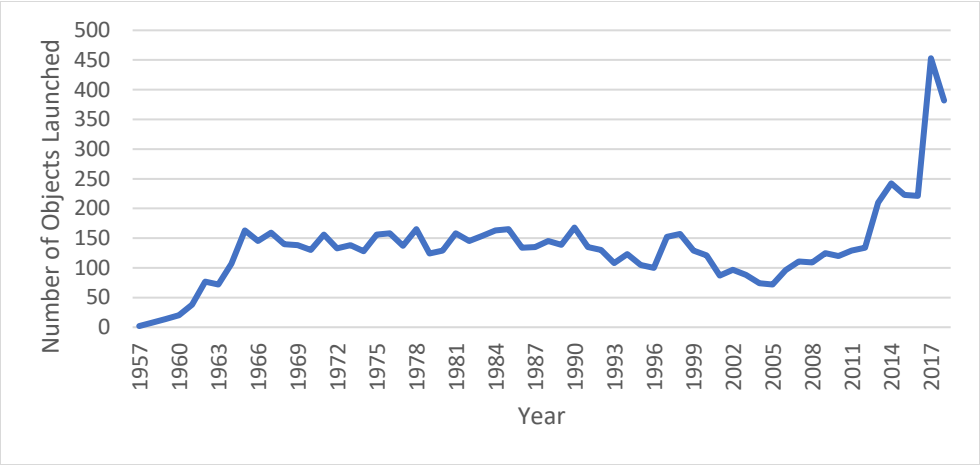


Figure 7.11 Objects launched into space each year. Data obtained from [115].

OneWeb has currently launched 6 satellites in 2019 [116] and aims to have a global constellation of 650 satellites by 2021 [117]. SpaceX's Starlink plans to launch 11,000 satellites into low Earth orbit by mid-2020 [118]. Both companies alone will exceed previous years launch totals.

With increased number of spacecrafts launched, the number of satellites that would experience on-orbit failures will also increase. There is a 3-4% chance that a spacecraft will experience a total or partial failure in the first 30 mission days [7]. The cumulative cost of these failures account for a total of $4.4 billion dollars in losses between the years of 1990 and 2006 [8]. A failed satellite can threaten the entire constellation and a servicing vehicle can be used to repair or remove the satellite from the constellation. The servicing vehicle can be contracted to de-orbit, repair, and refuel satellites regaining many of the costs that would be associated with the active debris removal activates.

This technology is becoming viable for both geostationary and low earth orbit satellites. In October of 2019, Northrop Grumman developed the Mission Extension Vehicle-1 servicing spacecraft which is designed to provide positioning for spacecraft that run out of fuel [119], as the spacecraft is fully functional. It can be more cost effective to re-position or refuel working geostationary satellites than to launch a replacement spacecraft. They were contracted to capture Intelsat-901 communications satellite launched in 2001, which is about to run out of fuel. MEV-1 will dock with the Intelsat-901 and extend its life for 5 years. After 5 years MEV-1 will either extend its contract with Intelsat-901 continuing to extend its life or move onto another satellite [119]. MEV-1 can deliver more than 15 years of life-extension service [63].

In December of 2019, OneWeb teamed up with Altius Space Machines to install their grappling fixture to their constellation to perform deorbiting as part of their responsible space initiative [35]. As the space industry is moving to a more responsible clean initiative, methods of deorbiting damaged or decommissioned satellites is vital and with servicing spacecraft, also provides the opportunity to connect and transfer data from the onboard computer of the decommissioned satellite before de-orbit.

Astroscale is another company focused on removing decommissioned or failed satellites. The company's overall mission is to make space more sustainable. Their goal is to work on end-of-life services for new satellites [119].

References

[1] P. Palmieri, S.P. Pastorelli, M. Scarcia, S. Mauro, Grasping Mechanism Concepts for Space Debris Removal Applications, (2019) 1–11.

[2] ESA and the United Nations team up for space debris, (2019). https://www.esa.int/Safety_Security/ESA_and_the_United_Nations_team_up_for_space_debris (accessed April 16, 2020).

[3] Clean Space, (n.d.). https://www.esa.int/Enabling_Support/Space_Engineering_Technology/Clean_Space (accessed December 10, 2019).

[4] N. Lee, C.-Y. Hou, How Europe Will Save Space Travel, 2018. https://www.youtube.com/watch?v=Y9qWS8hlUS4.

[5] ESA's e.Deorbit debris removal mission reborn as servicing vehicle, (2018). https://www.esa.int/Safety_Security/Clean_Space/ESA_s_e.Deorbit_debris_removal_mission_reborn_as_servicing_vehicle.

[6] This spacecraft will clean up space debris with its harpoon: About RemoveDebris experiment, (2018). https://www.indiatoday.in/education-today/gk-current-affairs/story/how-this-spacecraft-will-clean-up-space-debris-with-its-harpoon-1209006-2018-04-10 (accessed May 3, 2019).

[7] B.R. Sullivan, D.L. Akin, A survey of serviceable spacecraft failures, AIAA Sp. 2001 Conf. Expo. (2001) 1–8. doi:https://doi.org/10.2514/6.2001-4540.

[8] G.A. Landis, S.G. Bailey, R. Tischler, Causes of power-related satellite failures, Conf. Rec. 2006 IEEE 4th World Conf. Photovolt. Energy Conversion, WCPEC-4. 2 (2007) 1943–1945. doi:10.1109/WCPEC.2006.279878.

[9] K. Yoshida, Space Robot Dynamics and Control: To Orbit, From Orbit, and Future, Robot. Res. (2000) 449–456. doi:10.1007/978-1-4471-0765-1_54.

[10] A. Flores-Abad, O. Ma, K. Pham, S. Ulrich, A review of space robotics technologies for on-orbit servicing, Prog. Aerosp. Sci. 68 (2014) 1–26. doi:10.1016/j.paerosci.2014.03.002.

[11] K.H. Doetsch, G. Lindgerg, L. Neilson, E. James-abra, Canadarm, (2015). https://www.thecanadianencyclopedia.ca/en/article/canadarm (accessed January 16, 2020).

[12] G. Dong, Autonomous Visual Servo Robotic Capture of Non-Cooperative Target, (2016).

[13] R. Rembala, C. Ower, Robotic assembly and maintenance of future space stations based on the ISS mission operations experience, Acta Astronaut. 65 (2009) 912–920. doi:10.1016/j.actaastro.2009.03.064.

[14] Canadarm, Canadarm2, and Canadarm3 – A comparative table, (2019). https://www.asc-csa.gc.ca/eng/iss/canadarm2/canadarm-canadarm2-canadarm3-comparative-table.asp (accessed January 16, 2020).

[15] The Lunar Gateway, (2019). https://csa-asc.gc.ca/eng/astronomy/moon-

exploration/lunar-gateway.asp (accessed January 17, 2020).

[16] About Dextre, (2018). https://www.asc-csa.gc.ca/eng/iss/dextre/about.asp.

[17] P. Laryssa, E. Lindsay, O. Layi, International space station robotics: a comparative study of ERA, JEMRMS and MSS, … Robot. …. (2002) 1–8. http://robotics.estec.esa.int/ASTRA/Astra2002/Papers/astra2002_1.3-1.pdf.

[18] W.J. Li, D.Y. Cheng, X.G. Liu, Y.B. Wang, W.H. Shi, Z.X. Tang, F. Gao, F.M. Zeng, H.Y. Chai, W.B. Luo, Q. Cong, Z.L. Gao, On-orbit service (OOS) of spacecraft: A review of engineering developments, Prog. Aerosp. Sci. 108 (2019) 32–120. doi:10.1016/j.paerosci.2019.01.004.

[19] G. Hirzinger, ROTEX — The first space robot technology experiment, Exp. Robot. III. (2005) 579–598. doi:10.1007/bfb0027622.

[20] T.J. Debus, S.P. Dougherty, Overview and performance of the Front-End Robotics Enabling Near-Term Demonstration (FREND) robotic arm, AIAA Infotech Aerosp. Conf. Exhib. AIAA Unmanned...Unlimited Conf. (2009) 1–12. doi:10.2514/6.2009-1870.

[21] NASA, On-Orbit Satellite Servicing Study, 2010.

[22] A. K, E. David, A. Krolikowski, E. David, Commercial On-Orbit Satellite Servicing: National and International Policy Considerations Raised by Industry Proposals, New Sp. 2013. 1 (2013) 29–41. doi:10.1089/space.2013.0002.

[23] Nasa, In-Space Robotic Manufacturing and Assembly (IRMA) Update for NAC TI&E Committee, (2016). https://www.nasa.gov/sites/default/files/atoms/files/nac_tkortes_irma_nov2016_ta

gged.pdf.

[24] D. Werner, SSL aims to parlay NASA, DARPA work into viable in-orbit repair business, Sp. News. (2018). https://spacenews.com/ssl-aims-to-parlay-nasa-darpa-work-into-viable-in-orbit-repair-business/.

[25] Restore-L Robotic Servicing Technology, (n.d.). https://sspd.gsfc.nasa.gov/restore-L.html (accessed February 17, 2020).

[26] Restore-L: Proving Satellite Servicing, (n.d.).

[27] M. Jankovic, F. Kirchner, Trajectory generation method for robotic free-floating capture of a non-cooperative, tumbling target, Astrophys. Sp. Sci. Proc. 52 (2018) 111–127. doi:10.1007/978-3-319-69956-1_7.

[28] M. Shan, J. Guo, E. Gill, Review and comparison of active space debris capturing and removal methods, Prog. Aerosp. Sci. 80 (2016) 18–32. doi:10.1016/j.paerosci.2015.11.001.

[29] DEOS, (n.d.). https://space.skyrocket.de/doc_sdat/deos.htm (accessed April 20, 2020).

[30] T. Pultarova, European Space Junk Cleanup Concept Gets New Mission: Refuel and Repair, (2019). https://www.space.com/43157-e-deorbit-new-refuel-repair-mission.html (accessed February 17, 2020).

[31] IN-ORBIT SERVICING/ACTIVE DEBRIS REMOVAL, ESA. (n.d.). http://m.esa.int/Our_Activities/Space_Safety/Clean_Space/in-orbit_servicing_active_debris_removal.

[32] G. Hausmann, M. Wieser, R. Haarmann, A. Brito, J.-C. Meyer, S. Jäkel, M.

Lavagna, B. Jakobsson, R. Biesbroek, E.Deorbit Mission: OHB Debris Removal Concepts, 13th Symp. Adv. Sp. Technol. Robot. Autom. (2015). http://elib.dlr.de/100732/1/96014_Wieser.pdf.

[33] D. Szondy, Space fishing: ESA floats plan to net space junk, NEW ATLAS. (2014). https://newatlas.com/esa-clean-space-junk-edeorbit/30972/ (accessed February 17, 2020).

[34] Technologies of Altius, (n.d.). https://www.altius-space.com/technologies.html (accessed February 16, 2020).

[35] OneWeb Grappling Fixture, (n.d.). https://www.youtube.com/watch?time_continue=1&v=nTtskTGRr1U&feature=emb_logo (accessed February 16, 2020).

[36] Shuttle Technical Facts, (n.d.). https://www.esa.int/Science_Exploration/Human_and_Robotic_Exploration/Space_Shuttle/Shuttle_technical_facts (accessed January 17, 2020).

[37] Remy Melina, How Big Is the International Space Station?, (2010). https://www.livescience.com/32583-how-big-is-the-international-space-station.html (accessed January 17, 2020).

[38] ETS-VII, (n.d.). https://directory.eoportal.org/web/eoportal/satellite-missions/e/ets-vii (accessed May 19, 2019).

[39] Orbital Express Astro, Astronautix. (n.d.). http://www.astronautix.com/o/orbitalexpressastro.html (accessed January 26, 2019).

[40] T.M. Espero, Orbital Express, (2008). https://www.slideserve.com/patsy/orbital-express-a-new-chapter-in-space.

[41] Aaron Saenz, Dextre Robot On Space Station Almost Ready For Duty, (2010). https://singularityhub.com/2010/07/27/dextre-robot-on-space-station-almost-ready-for-duty/ (accessed April 20, 2020).

[42] C. Henry, DARPA Revamps Phoenix In-Orbit Servicing Program, Via Satell. (2015). https://www.satellitetoday.com/government-military/2015/06/02/darpa-revamps-phoenix-in-orbit-servicing-program/ (accessed January 26, 2019).

[43] G.H. PhD, B. Kelm, DARPA Phoenix, (2014). https://www.nasa.gov/sites/default/files/files/G_Henshaw-NRL_Advances_in_Orbital_Inspection.pdf.

[44] P. Rank, Q. Mühlbauer, W. Naumann, K. Landzettel, Deos Automation and Robotics Payload, Kayser-Threde GmbH Report, 2011. http://robotics.estec.esa.int/ASTRA/Astra2011/Presentations/session 4b/01_rank.pdf.

[45] DEOS PhaseA Datasheet.pdf, (n.d.). https://spacetech-i.com/images/products/mission_and_satellites/deos/DEOS_PhaseA_Datasheet.pdf.

[46] P. Rank, Q. Mühlbauer, W. Naumann, K. Landzettel, The DEOS Automation and Robotics Payload, 11th Symp. Adv. Sp. Technol. Robot. Autom. (2011) 1–8. http://robotics.estec.esa.int/ASTRA/Astra2011/Papers/04B/FCXNL-11A06-2139219-1-2139219rank.pdf.

[47] European Robotic Arm, (n.d.).

https://www.esa.int/Science_Exploration/Human_and_Robotic_Exploration/International_Space_Station/European_Robotic_Arm.

[48] H.J. Cruijssen, M. Ellenbroek, M. Henderson, H. Petersen, P. Verzijden, M. Visser, The European Robotic Arm: A High-Performance Mechanism Finally on its way to Space, (2014).

[49] W. Xu, B. Liang, C. Li, Y. Liu, Y. Xu, Autonomous target capturing of free-floating space robot: Theory and experiments, Robotica. 27 (2009) 425–445. doi:10.1017/S0263574708004839.

[50] W. Xu, C. Li, X. Wang, Y. Liu, B. Liang, Y. Xu, Study on non-holonomic cartesian path planning of a free-floating space robotic system, Adv. Robot. 23 (2009) 113–143. doi:10.1163/156855308X392708.

[51] B. Ma, Inverse Kinematics 1 1/29/2018, (2018). https://www.eecs.yorku.ca/course_archive/2017-18/W/4421/lectures/Inverse kinematics - annotated.pdf.

[52] CS 4733 Class Notes: Kinematic Singularities and Jacobians 1 Kinematic Singularities, (n.d.) 1–9. http://www.cs.columbia.edu/~allen/F15/NOTES/jacobians.pdf.

[53] B. Goodwine, J. Nightingale, The effect of dynamic singularities on robotic control and design, Proc. - IEEE Int. Conf. Robot. Autom. (2010) 5213–5218. doi:10.1109/ROBOT.2010.5509738.

[54] E. Papadopoulos, S. Dubowsky, Dynamic singularities in free-floating space manipulators, J. Dyn. Syst. Meas. Control. Trans. ASME. 115 (1993) 44–52.

doi:10.1115/1.2897406.

[55] Mission concept and the role of ATV, (n.d.). http://www.esa.int/Science_Exploration/Human_and_Robotic_Exploration/ATV/Mission_concept_and_the_role_of_ATV (accessed June 6, 2020).

[56] D. Szondy, ESA developing satellite navigation markers for easier docking in space, (2019). https://newatlas.com/esa-space-navigation-markers/59433/.

[57] Altius Space Missions, (n.d.). https://www.altius-space.com/.

[58] R. Biesbroek, T. Soares, L. Innocenti, The E . Deorbit Cdf Study : a Design Study for the Safe, 2013 (2013) 22–25.

[59] F.J. Morring, An End To Space Trash?, Aviat. Week. (n.d.).

[60] P.B. de Selding, Intelat Signs up for MDA's Satellite Refueling Service, Sp. News. (2011). https://archive.is/20120321160118/http://www.sbv.spacenews.com/satellite_telecom/110318intelsat-signs-for-mdas-satellite-refueling-service.html (accessed February 6, 2020).

[61] M.H. Kaplan, DYNAMICS AND CONTROL OF DETUMBLING A DISABLED SPACECRAFT DURING RES?CUE OPERATIONS, (2020).

[62] Loren Grush, SpaceX's Crew Dragon successfully docks with the space station, The Verge. (2020). https://www.theverge.com/2020/5/31/21271269/spacex-docking-iss-crew-dragon-nasa-success (accessed June 7, 2020).

[63] E. Howell, Two private satellites just docked in space in historic first for orbital servicing, Space.Com. (2020). https://www.space.com/private-satellites-docking-

success-northrop-grumman-mev-1.html (accessed March 23, 2020).

[64] Space Shuttle Canadarm Robotic Arm Marks 25 Years in Space, (n.d.). https://www.nasa.gov/mission_pages/shuttle/behindscenes/rms_anniversary.html (accessed June 7, 2020).

[65] Expedition 63 Awaits SpaceX Crew, Unpacks Japanese Cargo, (2020). https://blogs.nasa.gov/spacestation/tag/canadarm2/ (accessed June 7, 2020).

[66] E. Howell, What It's Like to Snag a Spacecraft with the International Space Station's Robotic Arm, (2018). https://www.space.com/41776-capturing-a-spacecraft-with-robotic-arm-canadarm2.html (accessed June 7, 2020).

[67] Dextre's Fuel Transfer task: Fill'er up, Robot!, (2019). https://www.asc-csa.gc.ca/eng/iss/rrm/task.asp (accessed June 7, 2020).

[68] M. Zebenay, T. Boge, R. Lampariello, D. Choukroun, Satellite Docking Simulator Based on Hardware-in-the-Loop Hybrid Contact Model, Int. Symp. Artif. Intell. Robot. Autom. Sp. - ISAIRAS. (2012).

[69] T. Rybus, K. Seweryn, J. Oleś, F.L. Basmadji, K. Tarenko, R. Moczydłowski, T. Barciński, J. Kindracki, Ł. Mężyk, P. Paszkiewicz, P. Wolański, Application of a planar air-bearing microgravity simulator for demonstration of operations required for an orbital capture with a manipulator, Acta Astronaut. 155 (2019) 211–229. doi:10.1016/j.actaastro.2018.12.004.

[70] N. Callens, J. Ventura-Traveset, T.L. De Lophem, C. Lopez De Echazarreta, V. Pletser, J.J.W.A. Van Loon, ESA parabolic flights, drop tower and centrifuge opportunities for university students, Microgravity Sci. Technol. 23 (2011) 181–

189. doi:10.1007/s12217-010-9181-1.

[71] T. Rybus, K. Seweryn, M. Ciesielska, K. Grassmann, J. Grygorczuk, M. Kowalski, M. Krzewski, J. Lisowski, K. Skup, T. Szewczyk, R. Wawrzaszek, New Planar Air-Bearing Microgravity Simulator for Verification of Space Robotics Numerical Simulations and Control Algorithms, Astra 2013. (2013).

[72] Neutral Buoyancy Simulator - Solar Max TestingTitle, (2013). https://www.nasa.gov/centers/marshall/history/gallery/msfc_iow_7.html (accessed October 22, 2019).

[73] J.V. Llop, J. V. Drew, R. Zappulla, M. Romano, Autonomous capture of a resident space object by a spacecraft with a robotic manipulator: Analysis, simulation and experiments, AIAA/AAS Astrodyn. Spec. Conf. 2016. (2016). doi:10.2514/6.2016-5269.

[74] R. Zappulla, J. Virgili-Llop, C. Zagaris, H. Park, A. Sharp, M. Romano, Floating spacecraft simulator test bed for the experimental testing of autonomous guidance, navigation, and control of spacecraft proximity maneuvers and operations, AIAA/AAS Astrodyn. Spec. Conf. 2016. (2016). doi:10.2514/6.2016-5268.

[75] M. Sabatini, M. Farnocchia, G.B. Palmerini, Design and tests of a frictionless 2D platform for studying space navigation and control subsystems, IEEE Aerosp. Conf. Proc. (2012) 1–12. doi:10.1109/AERO.2012.6187259.

[76] M. Sabatini, P. Gasbarri, G.B. Palmerini, Coordinated control of a space manipulator tested by means of an air bearing free floating platform, Acta Astronaut. 139 (2017) 296–305. doi:10.1016/j.actaastro.2017.07.015.

[77] E. Papadopoulos, I.S. Paraskevas, T. Flessa, the Ntua Space Robot Simulator : Design & Results, Power. (2007).

[78] E. Papadopoulos, I.K. Kaliakatsos, D. Psarros, Minimum fuel techniques for a space robot simulator with a reaction wheel and PWM thrusters, 2007 Eur. Control Conf. ECC 2007. (2007) 3391–3398. doi:10.23919/ecc.2007.7068723.

[79] J. Kernot, Adaptive Control of a Tendon-Driven Manipulator for the Capture of Non-Cooperative Space Targets, (2020). doi:10.2514/6.2020-2080.

[80] Dr. Henry, Control of Free-Flying Space Robot Manipulator System, 1988. https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19880018179.pdf.

[81] S. Jaekel, R. Lampariello, W. Rackl, M. De Stefano, N. Oumer, A.M. Giordano, O. Porges, M. Pietras, B. Brunner, J. Ratti, Q. Muehlbauer, M. Thiel, S. Estable, R. Biesbroek, A. Albu-Schaeffer, Design and operational elements of the robotic subsystem for the e.deorbit debris removal mission, Front. Robot. AI. 5 (2018) 1–20. doi:10.3389/frobt.2018.00100.

[82] J. Cookson, EXPERIMENTAL INVESTIGATION OF SPACECRAFT RENDEZVOUS AND DOCKING BY DEVELOPMENT OF A 3 DEGREE OF FREEDOM SATELLITE SIMULATOR TESTBED, 2019.

[83] J.L. Schwartz, M.A. Peck, C.D. Hall, Historical review of spacecraft simulators, Adv. Astronaut. Sci. 114 (2003) 405–423.

[84] E.G. Papadopoulos, Nonholonomic Behavior in Free-floating Space Manipulators and its Utilization, Nonholonomic Motion Plan. (1993) 423–445. doi:10.1007/978-1-4615-3176-0_11.

[85] H. Yao, Y. Wang, J. Cui, J. Bao, H. Yang, Z. Zhu, G. Zheng, Z.H. Zhu, Implementation of Three DoFs Small Satellite Ground Simulation System, 54th AIAA Aerosp. Sci. Meet. (2016). doi:https://doi.org/10.2514/6.2016-0697.

[86] I.S. Paraskevas, G. Rekleitis, K. Nanos, Space Robotics Technologies for On-Orbit Servicing Missions, (2019) 1–12.

[87] S. Garrido-Jurado, R. Muñoz-Salinas, F.J. Madrid-Cuevas, M.J. Marín-Jiménez, Automatic generation and detection of highly reliable fiducial markers under occlusion, Pattern Recognit. 47 (2014) 2280–2292. doi:10.1016/j.patcog.2014.01.005.

[88] Jean-Yves Bouguet, Camera Calibration Toolbox for Matlab, (n.d.). http://www.vision.caltech.edu/bouguetj/calib_doc/.

[89] Mongoose Embedded Web Server Library, (n.d.). https://github.com/cesanta/mongoose.

[90] I.S. Division, M. Field, Control of Free-Flying Control of Free-Flying, 1987. https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19880018178.pdf.

[91] H. Nakanishi, K. Yoshida, Impedance control for free-flying space robots -basic equations and applications-, IEEE Int. Conf. Intell. Robot. Syst. (2006) 3137–3142. doi:10.1109/IROS.2006.282334.

[92] A. Stolfi, P. Gasbarri, M. Sabatini, Parametric analysis of a controlled deployable space manipulator used for capturing a non-cooperative satellite, Proc. Int. Astronaut. Congr. IAC. 12 (2017) 7775–7787. doi:10.1016/j.actaastro.2018.04.028.

[93] S.I. Nishida, S. Kawamoto, Strategy for capturing of a tumbling space debris, Acta Astronaut. 68 (2011) 113–120. doi:10.1016/j.actaastro.2010.06.045.

[94] S.A.A. Moosavian, Dynamics and control of free-flying robots in space: A survey, IFAC Proc. Vol. 37 (2004) 621–626. doi:10.1016/s1474-6670(17)32047-5.

[95] Cintula, Petre, Fermüller, C. G., Noguera, Carles, Fuzzy Logic, Stanford Encycl. Philos. (Fall 2017 Ed. (n.d.). https://plato.stanford.edu/entries/logic-fuzzy/.

[96] What is “fuzzy logic”? Are there computers that are inherently fuzzy and do not apply the usual binary logic?, (1999). https://www.scientificamerican.com/article/what-is-fuzzy-logic-are-t/ (accessed June 11, 2020).

[97] What is Fuzzy Logic, (n.d.). https://www.mathworks.com/help/fuzzy/what-is-fuzzy-logic.html.

[98] Fuzzy-Logic Control, (n.d.). https://www.sciencedirect.com/topics/engineering/fuzzy-logic-control (accessed December 3, 2019).

[99] Sugeno Fuzzy Model, Res. Hubs. (n.d.). http://researchhubs.com/post/engineering/fuzzy-system/takagi-sugeno-fuzzy-model.html (accessed June 11, 2020).

[100] J.R. Wertz, Spacecraft Attitude Determination and Control (Astrophysics and Space Science Library 73), 1999. https://books.google.ca/books?id=crTwCAAAQBAJ&pg=PA273&lpg=PA273&dq=sample+cold+gas+thruster+force+profile&source=bl&ots=J6NucZMDs_&sig=

ACfU3U3fU6-EirIFsYTbgjf3eWj54X7n8A&hl=en&sa=X&ved=2ahUKEwi7vpf0gPvpAhX0l3IEHao5AhYQ6AEwDHoECAoQAQ#v=onepage&q=sample cold gas thruster force profile&f=false.

[101] K. Yoshida, R. Kurazune, Y. Umetani, Dual Arm Coordination in Space Free-Flying Robot, Proceedings. 1991 IEEE Int. Conf. Robot. Autom. (1991). doi:10.1109/ROBOT.1991.132004.

[102] C.B. Manipulators, J. Russakow, S.M. Rock, Extended Operational Space Formulation for Serial-to-Parallel Chain (Branching) Man, (n.d.) 1056–1061.

[103] J. Leitner, Multi-robot cooperation in space: A survey, Proc. - 2009 Adv. Technol. Enhanc. Qual. Life, AT-EQUAL 2009. (2009) 144–151. doi:10.1109/AT-EQUAL.2009.37.

[104] N.P. Lucas, A.K. Pandya, R.D. Ellis, Review of multi-robot taxonomy, trends, and applications for defense and space, Unmanned Syst. Technol. XIV. 8387 (2012) 83871N. doi:10.1117/12.919567.

[105] P. Boning, S. Dubowsky, Coordinated control of space robot teams for the on-Orbit construction of large flexible space structures, Adv. Robot. 24 (2010) 303–323. doi:10.1163/016918609X12619993300665.

[106] Andy, How many satellites orbiting the Earth in 2019?, Pixalytics. (2019). https://www.pixalytics.com/satellites-orbiting-earth-2019/ (accessed May 2, 2019).

[107] P.D. Anz-meador, J.N. Opiela, D. Shoots, HISTORY OF ON-ORBIT SATELLITE FRAGMENTATIONS 15 th Edition Orbital Debris Program Office,

2019.

[108] O. Debris, P. Office, HISTORY OF ON-ORBIT SATELLITE FRAGMENTATIONS 14 th Edition, (2008).

[109] M. Safi, H. Devlin, “A terrible thing”: India’s destruction of satellite threatens ISS, says Nasa, Guard. (2019). https://www.theguardian.com/science/2019/apr/02/a-terrible-thing-nasa-condemns-indias-destruction-of-satellite-and-resulting-space-junk (accessed December 10, 2019).

[110] O. Debris, P. Office, HISTORY OF ON-ORBIT SATELLITE FRAGMENTATIONS 13 th Edition Orbital Debris Program Office, (2004).

[111] R.S. Jakhu, Y.O.M. Nyampong, T. Sgobba, Regulatory framework and organization for space debris removal and on orbit servicing of satellites, J. Sp. Saf. Eng. 4 (2017) 129–137. doi:10.1016/j.jsse.2017.10.002.

[112] AGREEMENT AMONG THE GOVERNMENT OF CANADA, GOVERNMENTS OF MEMBER STATES OF THE EUROPEAN SPACE AGENCY, THE GOVERNMENT OF JAPAN, THE GOVERNMENT OF THE RUSSIAN FEDERATION, AND THE GOVERNMENT OF THE UNITED STATES OF AMERICA CONCERNING COOPERATION ON THE CIVIL, (n.d.). https://aerospace.org/sites/default/files/policy_archives/Space Station Intergovernmental Agreement Jan98.pdf.

[113] A.M. Long, M.G. Richards, D.E. Hastings, On-orbit servicing: A new value proposition for satellite design and operation, J. Spacecr. Rockets. 44 (2007) 964–976. doi:10.2514/1.27117.

[114] A.R. Graham, J. Kingston, Assessment of the commercial viability of selected options for on-orbit servicing (OOS), Acta Astronaut. 117 (2015) 38–48. doi:10.1016/j.actaastro.2015.07.023.

[115] Andy, How many satellites are orbiting the Earth in 2018?, Pixalytics. (2018). https://www.pixalytics.com/sats-orbiting-the-earth-2018/ (accessed May 3, 2019).

[116] H. Weitering, In Photos: OneWeb Launches New Global Satellite Internet Constellation, Space.Com. (2019). https://www.space.com/oneweb-1st-satellite-launch-photos.html (accessed May 2, 2019).

[117] OneWeb Secures $1.25 Billion in New Funding After Successful Launch, (2019). https://www.oneweb.world/newsroom/oneweb-secures-1-25-billion-in-new-funding-after-successful-launch (accessed May 2, 2019).

[118] J. Fingas, Elon Musk shows SpaceX's first internet satellites ready for launch, Engadget. (2019). https://www.engadget.com/2019/05/12/elon-musk-shows-spacex-internet-satellites/ (accessed May 14, 2019).

[119] Loren Grush, Next year, new space missions will test technologies to fix busted satellites in orbit, The Verge. (2019). https://www.theverge.com/2019/10/14/20909578/satellite-repair-servicing-missions-northrop-grumman-astroscale-space-debris (accessed March 23, 2020).