Design and Analysis of a Estimation and Control Pipeline for a Robot Manipulator in Space

Daniel Vedova

CMU-RI-TR-21-63

August 12 2021

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

To mom and dad.

Abstract

Many satellites are rapidly reaching the end of their lifespans, and risk de-orbiting if no action is taken. One common problem satellites face towards the end of their lifespans is that they are running out of fuel, therefore new propulsion units must be delivered in-orbit. Attempting to deliver propulsion units using human astronauts is both dangerous and cost-prohibitive. To meet this challenge, corporate entities such as Northrop Grumman have led efforts to develop cost-effective, robotic in-orbit satellite servicing vehicles capable of delivering life extension payloads to satellites in need of maintenance. The realization of robotic deliveries of life extension payload is the beginning of a larger effort to perform more general on-orbit maintenance tasks using robotic tools.

In this thesis we present an integrated tracking, estimation, and control framework for space robots, along with an environment to simulate in-orbit satellite servicing missions. We show that existing methods for operational space control of floating base manipulators can be extended to partially observable environments by incorporating contact force information into the estimation and control problem.

For our control subsystem, we first evaluate multiple control solutions (adaptive model predictive control, nonlinear model predictive control, operational space control). After evaluating both optimization-based approaches, (model predictive control) and classical approaches (operational space control), we selected an operational space control method to send control commands. We selected this controller because we concluded that optimization-based methods struggle to run in real time for our high degree of freedom system. We present two types of analysis for our controller: feasibility, and stability analysis. We perform multiple types of analysis to demonstrate the efficacy of our controller. We perform feasibility analysis in order to numerically compute the space of initial robot end-effector poses for which the controller can successfully complete a docking mission. We perform stability analysis to guarantee the stability of our controller using analytical methods from non-linear control theory.

We analyze our vision tracking subsystem by conducting a series of experiments to understand the configurations in which our vision subsystem succeeds and fails, and with what degree of confidence it reports measurements of the client satellite. We focus on understanding how the motion of our robot arm and the end-effector payload effects the performance of the tracker through occlusions of the client satellite, and we search for configurations of our robot base and arm which minimize occlusions. This analysis enables us to better position the arm and the MEP during docking operations.

Acknowledgments

If I learned anything from six years at Carnegie Mellon, it is the following: No one accomplishes anything of significance alone. With that being said, I would like to start by thanking the Department of the Navy for giving me the opportunity to delay my training as a Submarine Warfare Officer in order to pursue my master's studies. In particular, I would like to thank the command at the CMU Naval ROTC Unit for supporting my studies and my transition from a midshipman to a Naval Officer.

I am incredibly grateful to Professors Howie Choset and Matt Travers. Over the the past six years, they've supported me academically, professionally, and personally. Howie and Matt: I'm honored to have studied under your supervision during this transformative period of my life.

Thank you to the Northrop Grumman Corporation for supporting the Biorobotics Lab, and giving me the opportunity to work on this project. In particular, I'd like to thank Dave Bodkin and his team at NGC Mission Systems for sharing their insights and practical advice with me.

During my time in the Biorobotics Lab, I've had the opportunity to work with many brilliant students, staff, and post-docs. I'd like to thank Ian Abraham and Vitaliy Fedonyuk for being great friends and even better post-docs. I'd like to thank some of my peers for helping me along during my studies: Jaskaran Grover, Haowen Shi, Ben Freed, Julian Whitman, Matt Martone, Shuo Yang, Hans Kumar, Raunaq Bhirangi, Alex Bouman, Kirtan Patel, Raghavv Goel, Anna Yu, Zheng Xu, Emma Benjaminson, David Neiman, Abhimanyu, Anoop Bhat, Chase Noren, Mickey Velado, Steve Crews, David Beskow, Tejas Zodage, Kevin Tracy, and Tommy Moriarty. Several staff members have also played a key role in my development as a researcher, and also as a person: Peggy Martin, B.J. Fecich, Lu Li, Jim Picard, Chuck Whittaker, Tim Angert, Ben Brown, Jean Harpley, and Dennis Fortner.

I owe a great deal to my closest friends who have stuck by me through this process: Stefan Zhu, Evan Glauberman, and Ray Galeza.

The last group of people who deserve recognition is my wonderful family. I am forever grateful to my girlfriend, Maddie, my brother, Patrick, my cousin Brucie, and to my mother and father, for their unwavering support over the years. Mom and Dad: Thank you for everything you did to get me this far. I wish only to continue to make the two of you proud.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

There exist dozens of aging satellites in orbit currently at risk of de-orbiting due to a lack of propulsion [26]. Collectively, these satellites are worth billions of dollars and provide key services to society, such as weather information, GPS services, military intelligence, communications, and broadband internet services. Governments and companies who own these satellites have been spending growing amounts of resources to find alternatives to launching replacement satellites. This has led companies such as the Northrop Grumman Corporation (NGC) to develop in-orbit satellite-servicing assets. NGC developed the Mission Robotic Vehicle (MRV) to deliver Mission Extension Payloads (MEPs) to aging satellites in Low Earth Orbit (LEO). These MEPs are designed to provide auxiliary power and propulsion capabilities to the aging satellite running low on power and propellant. This thesis seeks to develop a simulated model of the docking operations, and develops a robust force-compliant control system to prevent error and minimize disturbances during docking.

1.2 Problem and Proposed Solution

The space environment we deal with consists of three separate bodies: The MRV, the MEP, and the Client satellite, or client vehicle. Recall that the MRV delivers

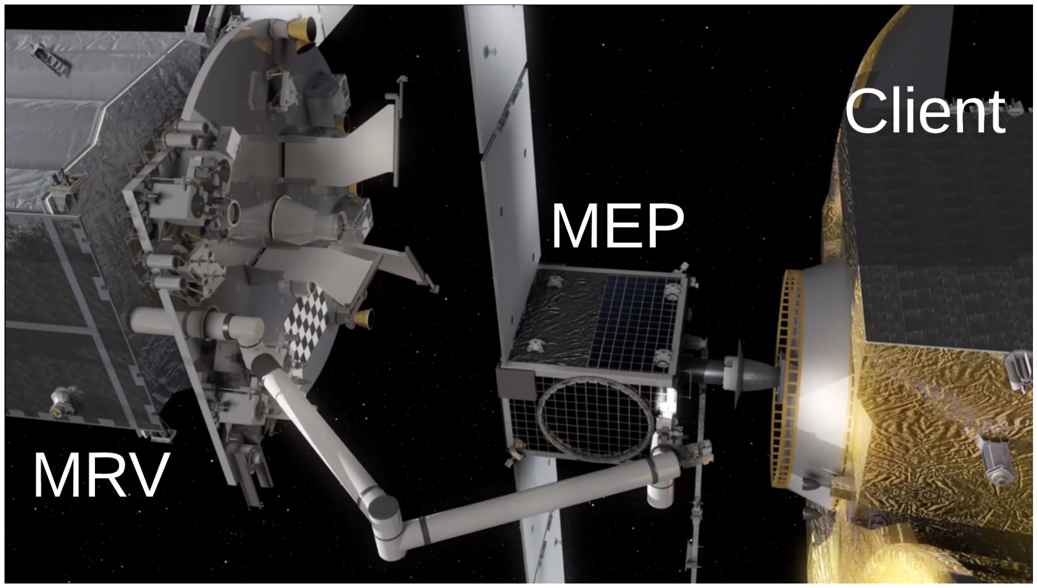


Figure 1.1: Artists Rendering: The NGC Mission Robotic Vehicle (MRV) delivering a Mission Extension Payload (MEP) to an aging client satellite. (Image courtesy of NGC)

the MEP to the client vehicle. Our satellite servicing mission consists of two phases: approach followed by docking. Fig. 1.6 provides a visual overview of the three key bodies interacting in our mission.

The problem we seek to solve is as follows: given an non-compliant[1] client satellite, the MRV must maneuver to attach a MEP to the rocket nozzle of the client satellite while minimally disturbing the operations of the client satellite. We can draw a parallel between our problem of driving an MEP towards a client satellite in space to a well-known problem in robotics: The problem of inserting a peg into a hole using a robot arm [16]. There are several challenges associated with controlling the MRV to successfully dock the MEP with a client satellite in space:

1. Floating base dynamics and controls

   In traditional manipulation problems, such as the peg-in-hole problem, where the base of the manipulator arm is fixed to the ground, one can leverage well-established manipulator dynamics and controls algorithms to move the

[1]In this context we define a *non-compliant* body as a body that is not under our control and possibly following commands from an unobserved controller.

end-effector of a manipulator to a desired location [4]. For in-orbit satellite servicing, there is no ground that can withstand the reaction forces induced by the motion of the manipulator. Even the MRV base thrusters and reaction wheels would not be able to counteract such reaction forces. This is because the MRV base thrusters have limited fuel capacity, and the reaction wheels on the base can risk saturating, and be unable to respond to applied torques. Additionally, the current operational protocol calls to disable the base thrusters and reaction wheels during servicing. The MRV base thrusters are turned off as a safety precaution during docking operations to minimize the maximum velocity of a collision if one were to occur. Since the base that our manipulator arm is mounted to is free floating, we must leverage control methods which account for the coupled dynamics that exist between our manipulator arm and the floating base to which it is attached. To accomplish this, we make use of Operational Space Control [22], which has been commonly used in legged robots to perform tasks in the work-space.

2. Tracking a partially occluded spacecraft

   To send accurate position estimates to our operational space controller, we need to have access to the precise location of the client satellite. To achieve this, we make use of a camera mounted on the base of our MRV to publish images of the client satellite which are fed into an off-the-shelf vision tracking algorithm [15] to detect the client satellite, and return the client satellite's measured pose with respect to the MRV. This tracking routine is complicated by the fact that the MEP partially obscures the camera's view of the client satellite. This obscuring of the client satellite introduces the common problem in computer vision known as occlusion. See Figs. 1.2 and 1.3. To combat the uncertainty that occlusion brings into our tracking measurement, we make use of a Hybrid Extended Kalman Filter (H-EKF) [12].

3. Minimizing applied disturbance forces to client satellite

   The client vehicle must not be displaced any more than 2 cm in translation, and no more than 0.2 degrees in rotation. For the example of a telecommunications satellite, these requirements ensure that the signals being sent back to Earth by the client satellite stay on target, and reach their desired destination. In order

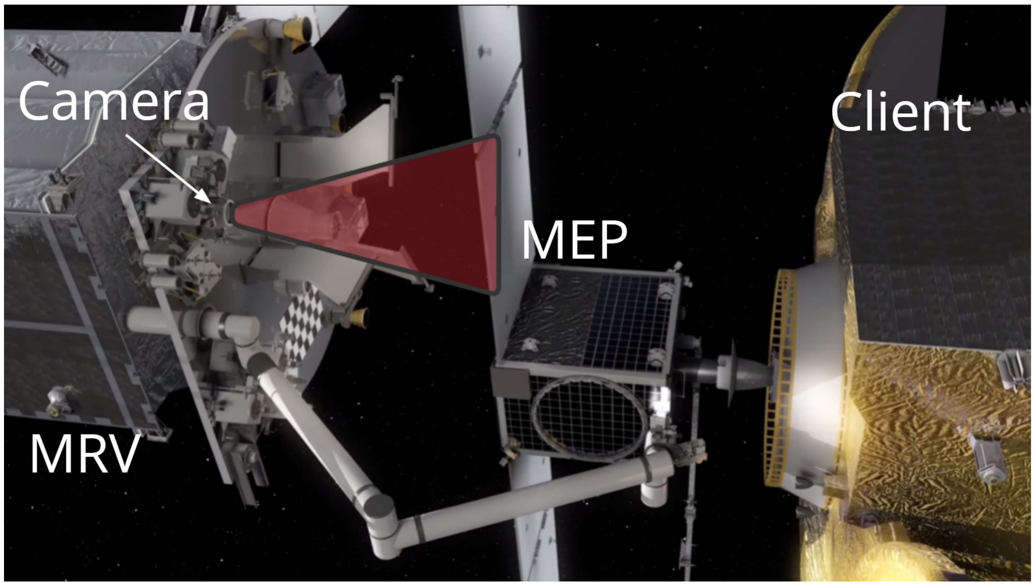


Figure 1.2: When mounted on the MRV's end-effector, the MEP partially occludes the camera tracker's view of the client satellite.

to comply with this requirement, we add a force compliant term in our controller to respond and react to any contact forces. See equation 3.39 in the Operational Space Controller section. This force compliance is crucial in scenarios with heavy occlusion, where the vision subsystem cannot see the client satellite perfectly. In a degraded vision-tracking scenario, force compliance allows the robot arm to react with the force and move the arm in such a way that future contact forces are reduced in cases where there is increased uncertainty about the position of the client satellite.

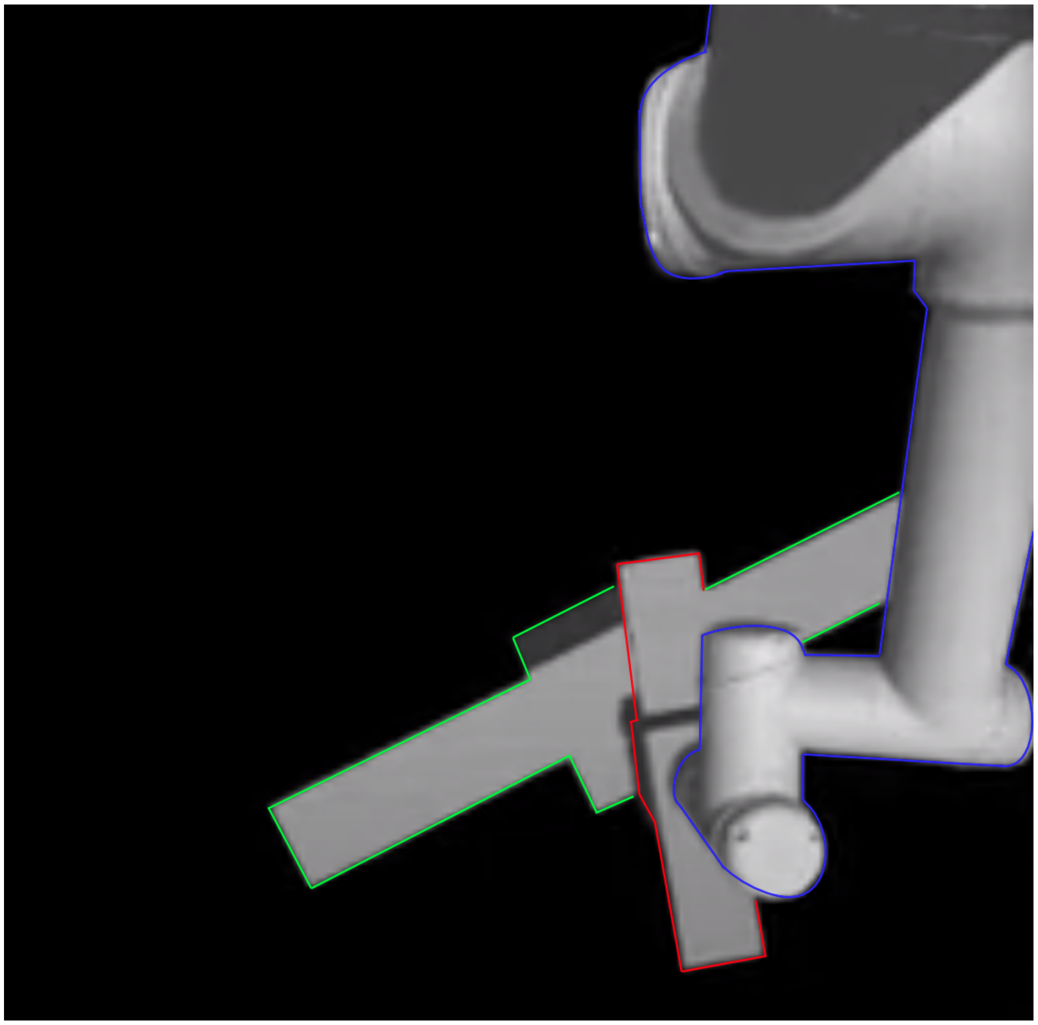


Figure 1.3: Example of occlusion from simulated MRV base camera feed. The client satellite (green) is partially occluded by the MRV arm (blue) and the MEP (red).

1.3 Mission Robotic Vehicle Control System Overview

The contributions we made to the MRV satellite systems are: A vision sensor/tracking system, a Hybrid Extended Kalman Filter (H-EKF), and an Operational Space Controller (OSC). As can be seen in Fig. 1.5, the control flow begins when the vision and force sensors (blue blocks in Fig. 1.5) take measurements of the environment (plant). These measurements (6-DoF force measurements and camera images) are passed into our H-EKF, and our vision tracking system, respectively (red blocks in Fig. 1.5). The output of the H-EKF represents the best estimate of the current pose of the client vehicle, and the current pose of the MRV end-effector (the MEP). These outputs are then subtracted from one another to generate the desired pose command, or error signal, to be sent to our Operational Space Control (OSC) system. By sending this error signal into the OSC system, the MEP will be driven into the client vehicle's rocket nozzle. The task block in Fig. 1.5 sends a desired pose signal. There are two desired poses: one for the approach phase, and one for the docking phase. The task block detects which phase the simulation is in, and then sends the corresponding signal for x_d.

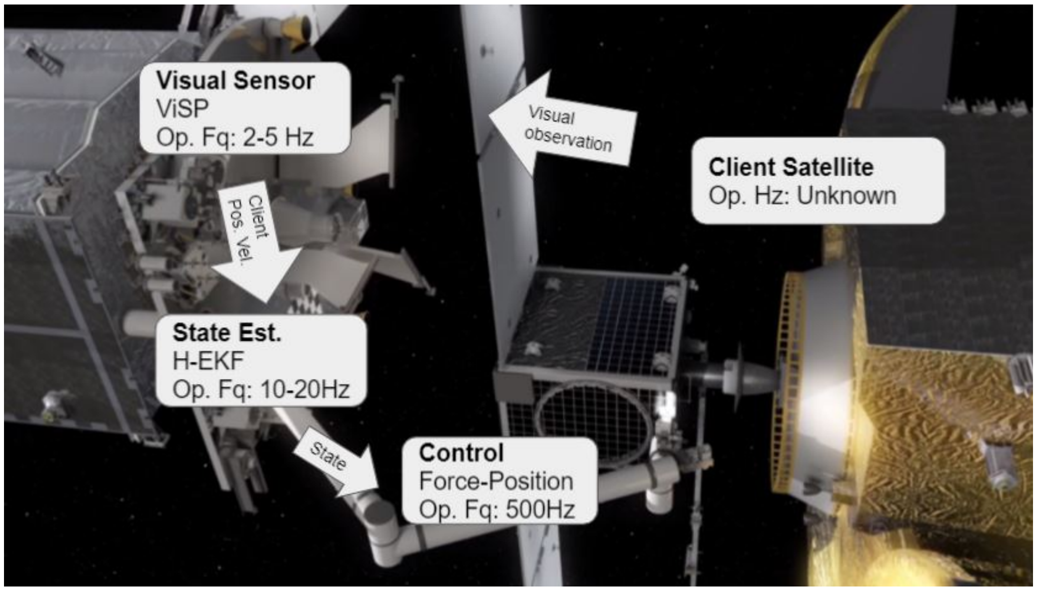


Figure 1.4: Summary of MRV control/communication flow. Nominal operating frequencies are labeled below each component.

1.3.1 Approach

The mission starts in the approach phase. In this phase of the operation, the MRV begins some distance away from the client satellite and uses its base thrusters and reaction wheels to move into docking position. The docking position is selected to be directly in front of the MEP, aligned along the axis of symmetry of the client vehicle's rocket nozzle. The MRV uses our Operational Space Controller to drive the MEP

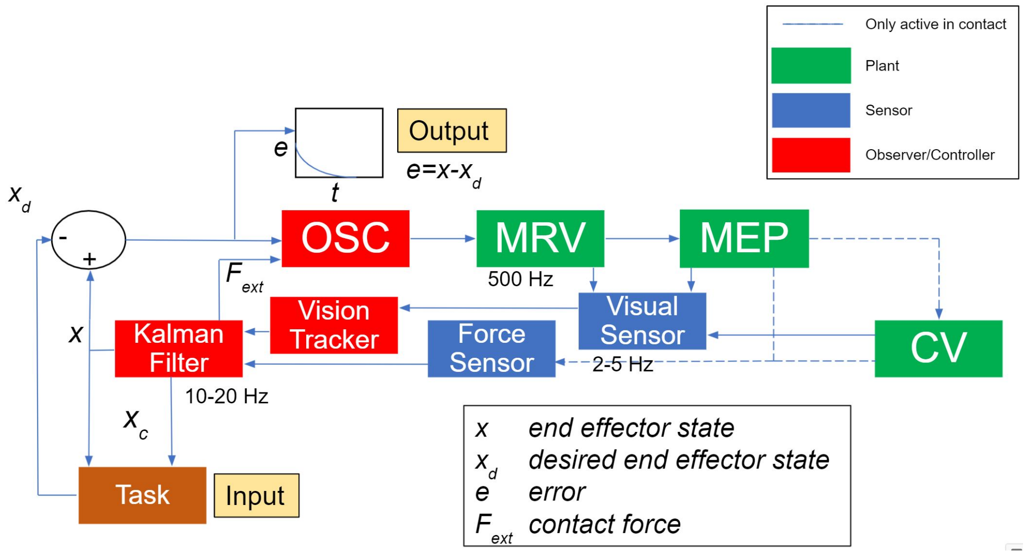


Figure 1.5: Block diagram of full MRV control system. Green blocks denote the plant, blue blocks denote sensors, red blocks denote controllers/observers. Solid lines represent signals that are active all the time, and dashed lines represent signals that are only active when an external forces is imparted onto the MRV. Note that the external force applied to the MEP, F_{ext} is passed into the OSC block in order to allow the OSC block to compute compliant controls that are aware of contact forces, as shown in equation 3.39.

to the desired docking state, directly in front of the client vehicle. Once the MRV reaches the docking position, the system enters the docking phase.

1.3.2 Docking

At this point, using the estimated position of the client vehicle coming from our vision tracker, we use our Operational Space Controller to move the MEP into the rocket nozzle of the client vehicle. The controller also makes use of the force sensor mounted in the end-effector of our robot manipulator to comply with any external forces applied to our arm. External forces measured by the MRV wrist force sensor are passed into the OSC controller, and used to guide the motion of the MEP away from the contact. The docking phase ends when the MEP reaches within 5mm of the final goal location inside of the client vehicle's rocket nozzle.

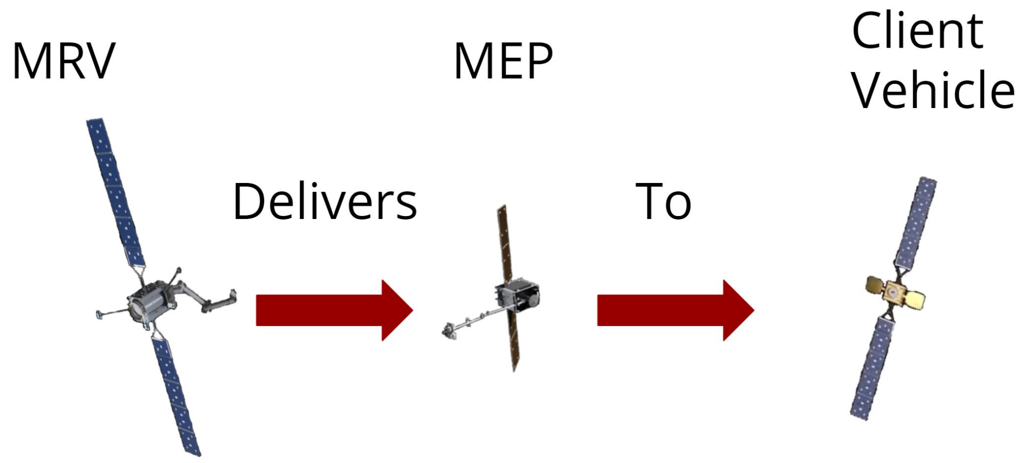


Figure 1.6: Overview of three key bodies in environment. The MRV (left) flies in low Earth orbit, acquires a MEP (center), and delivers it to an aging client vehicle (right). To be precise, all simulated MRV operations featured in our work assume that the MRV has already gone through the process of acquiring the MEP.

1.4 Contribution

The contribution of this work is two-fold: First, we present a simulated satellite servicing pipeline for the purposes of testing integrated solutions to the problem of docking with a non-compliant satellite in orbit. Secondly, we present our integrated tracking, estimation, and control system which enables a satellite with a manipulator arm to place the MEP in position with a non-compliant satellite in need of service. To demonstrate the efficacy of our proposed automated satellite docking system, we present individual component analyses of our vision tracker, H-EKF, and operational space controller, as well as system level analyses of our integrated system. The novelty of this pipeline is that we are leveraging both force and vision feedback to achieve our goal, in addition to also leveraging force feedback to improve state estimation of the pose of the client vehicle.

Chapter 2

Prior Work

There exists a great body of work for space robots, or floating base manipulator systems. Here, we break the prior work down into four subsections: a history of space robots, floating base manipulator kinematics and dynamics, control for floating base manipulators, and vision tracking in robotics.

2.1 Brief History of Robotic Space Systems and Floating Base Systems

There are several space launched robotic manipulator arms systems that have come before the NGC MRV system. One of the first robotic manipulators was launched into orbit when the Shuttle Remote Manipulator System (SRMS), which is also known as the Canadarm, was first deployed in the 1980s [27]. The Canadarm is a six degree of freedom robot arm that has been used to deploy satellites, assist with construction tasks, and assist astronauts with space walks. In the late 1990's, the Japanese government launched the Engineering Test Satellite (EST) VII experiment vehicle [18]. This space robot collected data for several docking and rendezvous missions. In the early 2000's, DARPA began a program called Orbital Express, whose goal was to develop a robotic satellite capable of servicing disabled satellites in orbit, much like the NGC MRV mission [2]. Images of these robotic space systems can be found in Fig. 2.1 Northrop Grummans's Mission Extension Vehicle (the predecessor to the

Mission Robotic Vehicle) completed its first servicing mission in orbit in Februrary 2020.

As an aside, we also draw inspiration from other floating base systems, such as microrobots designed to swim in in-viscid fluids, such as the system designed in [9] and [10]. Other systems with similar behavior to our system is the Chaplygin beanie, which was analyzed in work done by [1].

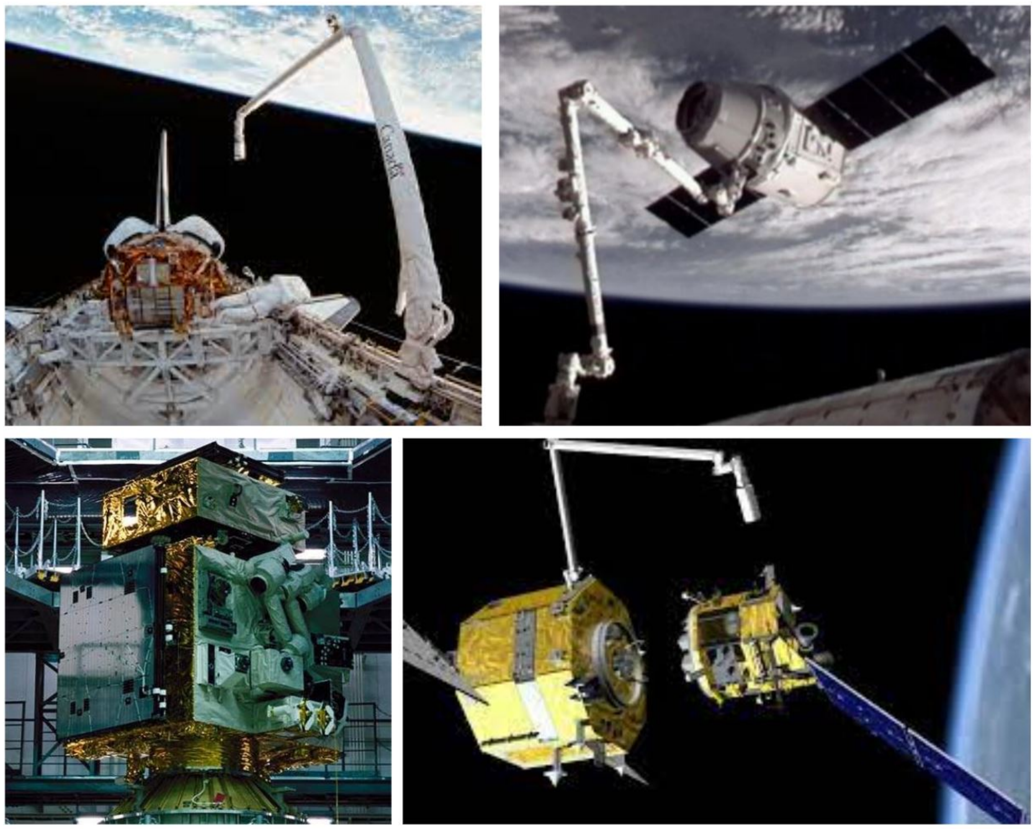


Figure 2.1: Top Left: Canadarm operating on a space shuttle mission. Top Right: Canadarm operating on the International Space Station. Bottom Left: Japanese ETS VII Experiment Vehicle. Bottom Right: Artist's rendering of DARPA Oribtal Express.

2.2 Floating Base Manipulator Kinematics and Dynamics

The theory behind floating base manipulator kinematics and dynamics was developed in the late 1980s and early 1990s. In 1993, engineers and researchers in the Spacecraft Engineering Department at the Naval Research Laboratory [14] [13]. In this work, the authors computed the forward kinematics of a generalized floating base manipulator, and also developed an inverse kinematics solution for a floating base robot with a six degree of freedom manipulator arm. They additionally analyzed the feasible work-space of a floating space robot with an unactuated base. [24] [31] both derived dynamic equations of motion for a six degree of freedom arm attached to a floating base. In their work, they also derived motion planning and control routines for space robots which brought the end effector of the arm from point A to point B. Fig. 2.2 shows a schematic for a generic space robot. In these works, the equations of motion for a generic space robot are derived by taking the following steps:

First, define the state of the system in vector form:

$$X = \begin{bmatrix} x_{base} \\ q \end{bmatrix} \tag{2.1}$$

Where x_{base} is the vector representing the six degree of freedom pose of the base and q is the vector representing the joint angles of the robot arm.

Next, write out the kinetic energy of the system:

$$T = \frac{1}{2} \begin{bmatrix} \dot{x}_{base}^T \dot{q}^T \end{bmatrix} A_{sys} \begin{bmatrix} \dot{x}_{base}^T \\ \dot{q}^T \end{bmatrix} \tag{2.2}$$

Where A_{sys} is the system mass matrix, which is found following the derivation from [28]:

We define A_{sys} as the full system mass matrix, which is a block matrix consisting of the base mass matrix, A_{11}, the manipulator arm mass matrix, A_{22}, and the mass matrices representing the coupling between the arm and the base, A_{21}, and A_{12}.

$$A_{\text{sys}} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \tag{2.3}$$

where

$$A_{11} = \begin{bmatrix} m_{\text{tot}} I_{3\times 3} & -m_{\text{tot}} \hat{r}_{\text{oc}} \\ m_{\text{tot}} \hat{r}_{\text{oc}} & H_{\text{s}} \end{bmatrix} \tag{2.4}$$

m_{tot} is the total mass of the system. $\hat{r}_{\text{oc}}$ is defined in Fig. 2.2.

H_{s} is given by

$$H_{\text{s}} = \sum_{i=1}^{6} \left(I_i - m_i \hat{r}_{0_i} \hat{r}_{0_i} \right) + I_0 \tag{2.5}$$

Where I_i is the inertia of the ith link of the MRV arm, m_i is the mass of the ith link of the arm, and $\hat{r}_{0_i}$ is defined in Fig. 2.2.

We can obtain the coupling inertia matrices with the following relationship:

$$A_{12} = \begin{bmatrix} J_{\text{TS}} \\ H_{\text{sq}} \end{bmatrix} \tag{2.6}$$

where J_{TS} is given by:

$$J_{\text{TS}} = \sum_{i=1}^{6} \left(m_i J_{T_i} \right) \tag{2.7}$$

J_{T_i} is found as follows:

$$J_{T_i} = \begin{bmatrix} \hat{k}_1 \left(r_i - P_1 \right) & \dots & \hat{k}_i \left(r_i - P_i \right) & 0_{3\times 6-i} \end{bmatrix} \ \forall \ (1 \leq i \leq 6) \tag{2.8}$$

$\hat{k}_i$ is the unit vector representing the axis of rotation of the ith joint of the MRV arm.

P_i is the position vector pointing from the inertial frame to the ith joint of the

MRV arm. r_i is the position vector pointing from the inertial frame to the center of mass frame of the ith link of the MRV arm.

And J_{R_i} is obtained with:

$$J_{R_i} = \begin{bmatrix} \hat{k}_1 & ... & \hat{k}_i & 0_{3\times 6-i} \end{bmatrix} \ \forall \ (1 \leq i \leq N) \tag{2.9}$$

We define our second coupling matrix by transposing A_{12}:

$$A_{21} = A_{12}^T \tag{2.10}$$

Lastly, we obtain the manipulator arm mass matrix with the following calculations:

$$A_{22} = \sum_{i=1}^{6} \left(J_{R_i}^T I_i J_{R_i} + m_i J_{T_i}^T J_{T_i} \right) \tag{2.11}$$

Now we can compute the Lagrangian (neglecting the potential term since we can neglect gravity effects in orbit):

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{x}_{base}}\right) - \frac{\partial T}{\partial x_{base}} = 0 \tag{2.12}$$

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}}\right) - \frac{\partial T}{\partial q} = \Gamma \tag{2.13}$$

Where Γ is a vector of manipulator arm joint torques.

This provides us with the equations of motion for our generic floating base manipulator system:

$$A_{sys} \begin{bmatrix} \ddot{x}_{base} \\ \ddot{q} \end{bmatrix} + b_{sys} = \begin{bmatrix} 0 \\ \Gamma \end{bmatrix} \tag{2.14}$$

Where b_{sys} is the vector of system Coriolis terms, defined below:

$$b_{\text{sys}} = \begin{bmatrix} \dot{A}_{11} & \dot{A}_{12} \\ \dot{A}_{21} & \dot{A}_{22} \end{bmatrix} \begin{bmatrix} \dot{x}_{\text{base}} \\ \dot{q} \end{bmatrix} + \begin{bmatrix} c_{\text{base}} \\ c_{\text{arm}} \end{bmatrix} \tag{2.15}$$

where

$$c_{\text{base}} = -\frac{1}{2}\frac{\partial}{\partial x_{\text{base}}} \left(\dot{x}_{\text{base}}^T A_{11} \dot{x}_{\text{base}} + \dot{q}^T A_{22} \dot{q} + \dot{x}_{\text{base}}^T A_{12} \dot{q} + \dot{q}^T A_{21} \dot{x}_{\text{base}} \right) \tag{2.16}$$

and

$$c_{\text{arm}} = -\frac{1}{2}\frac{\partial}{\partial q} \left(\dot{x}_{\text{base}}^T A_{11} \dot{x}_{\text{base}} + \dot{q}^T A_{22} \dot{q} + \dot{x}_{\text{base}}^T A_{12} \dot{q} + \dot{q}^T A_{21} \dot{x}_{\text{base}} \right) \tag{2.17}$$

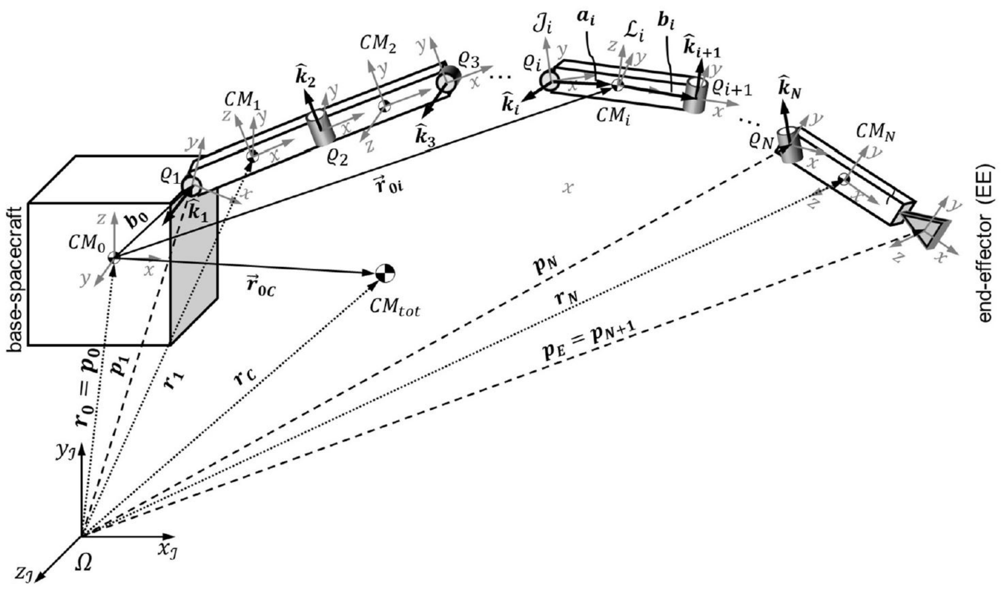


Figure 2.2: Kinematic diagram of a generic space robot. Image from [28]

2.3 Control for Floating Base Manipulators

There exists a variety of literature regarding the subject of controls for floating base manipulators. In 1987, Khatib et al developed a method called Operational Space Control (OSC) for fixed base manipulators [11]. The OSC formulation enables an engineer to determine generalized forces that act in the joint space (joint torque commands), based on a desired force to be applied in the work-space [11]. Khatib, along with Sentis et al. [22] extended Operational Space Control for use with floating base systems. Their use case was on a humanoid robot. See Fig. 2.3. In chapter 3, we will present our implementation of the Khatib OSC formulation, which includes modeling external forces into our computed torque computations. Several other control methods exist for floating base robots, such as resolved motion rate control, introduced by Yoshida and Umetani [31], and inverse dynamics control [20].

2.4 Vision Tracking

There are several solutions available to solving vision tracking problems. There are model-based approaches such as ViSP [15], and learning based approaches such as DOPE [23] and PoseCNN [30]. Other methods include particle-filter model-based tracker such as the Depth Based Object Tracking Library (dbot) [29], and search based methods such as PERCH from the SBPL at CMU [3]. Each one of these methods has pros and cons. Learning based methods require large quantities of training data in order to obtain high performance. Search based methods require onboard GPUs in order to guarantee real time functionality. Model-based methods require an accurate CAD model of the object to be tracked. Many of these methods also require a depth image to be provided in addition to an RGB image. We decided to use the ViSP model-based tracker because it does not require a depth image input, and it does not require the creation of a large training dataset. The reason we chose to avoid methods which rely on depth images is because NGC advised us that their proprietary vision tracking routine does not use depth information, and NGC recommended using a method that also does not use depth information, in order to more accurately represent and simulate the true NGC MRV system. Table 5.1 summarizes the pros and cons of each of these methods.

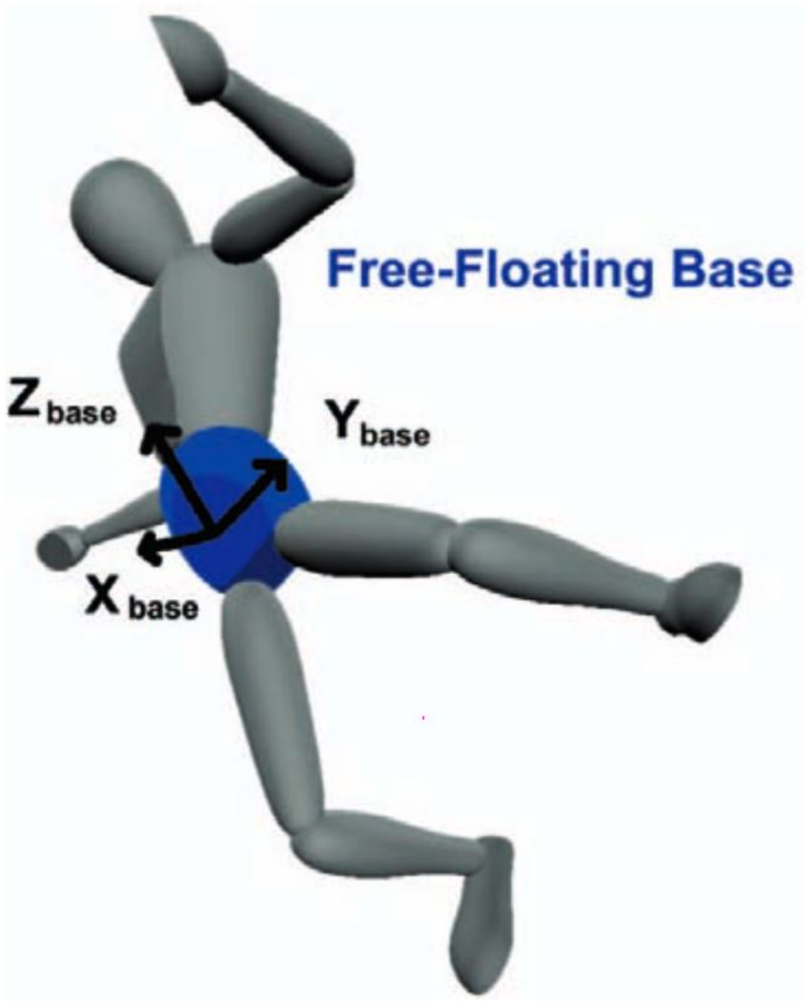


Figure 2.3: [22] developed an OSC formulation for floating base systems. Their test system was the above humanoid model. For their application, the operational space frame is the base frame located in the hip link of the humanoid robot. So for the humanoid OSC, joint torque commands in the legs and arms are computed to achieve a desired behavior in the base frame. For our space robot work, we define the operational space frame to be the end effector of the robot arm instead of the base link. Image from [22].

Tracking Method Comparison				
Method	Needs GPU?	Needs Training Data?	Needs CAD model?	Needs Depth Image?
ViSP	x	x	✓	x
DOPE	x	✓	x	x
PoseCNN	x	✓	x	x
dbot	x	x	✓	✓
PERCH	✓	✓	✓	✓

Table 2.1: Pros/cons of various types of vision trackers.

Chapter 3

Controller for Space Robot

3.1 Model Predictive Control for Planar Space Robot

As discussed in the prior work section, there are many different control methods available for use with floating base manipulators. Model Predictive Control (MPC) is a popular optimization-based method that has been implemented recently for operation on space robots. In this section, we explore the use of several classes of MPC controllers: Nonlinear MPC [21] and Adaptive MPC [6] on a planar space robot simulated in MATLAB.

The work by Tomasz Rybus et. al. in [21] explores maneuvers of a floating base satellite with an open chain manipulator to interact with a separate moving body. The study computes open-loop motion plans using trajectory optimization, and then treats those computed trajectories as references to track using NMPC and adaptive MPC. Our work builds a re-creation of the methods and results presented in the previous work, and explores an alternative MPC control method applied to the same motion scenarios. In implementing NMPC adaptive MPC for a planar space robot, we seek to understand the capabilities of NMPC and adaptive MPC for simplified floating base manipulators, as well as the limitations of NMPC/adaptive MPC.

To implement and explore the proposed paper of study [21], a representative system model was created using the Space Robotic Toolbox (SPART) toolbox [25]. The model provides us with a state dynamics description of a floating base manipulator system, this description provides us our system mass matrix and Coriolis terms. Trajectory optimizations were performed on the models, including a representation of the rendezvous action presented in the work. Differing instances of model predictive control (adaptive MPC and nonlinear MPC) were then applied to the generated trajectory, and compared both with and without model discrepancies[1].

Our motivation in implementing both NMPC and adaptive MPC for this system is to determine the pros and cons of each method, and to see which method, if any, is most suitable to serve as the controller for our floating base system.

Floating Base Dynamics for MPC

Here we discuss the dynamics of a floating base manipulator. Because the base of the robot is free floating, we must include the position and orientation of the base in the state description, as can be seen in equation 3.1. We modeled our system without an actuated base, and so our system is trivially underactuated. The floating base nature of our system makes computing the dynamics of the system more complicated, because our system must obey conservation of angular and linear momentum laws. Our system is governed by the following equations of motion:

$$\begin{bmatrix} H_b & H_c \\ H_c^T & H_m \end{bmatrix} \ddot{q} + \begin{bmatrix} c_b \\ c_m \end{bmatrix} = \Gamma \tag{3.1}$$

Where H_b is the inertia matrix of the base, H_c is the coupling inertia matrix between the base and the manipulator, and H_m is the manipulator inertia matrix. The coupling inertia matrix helps describe how the motion of the base affects the motion of the arm, and how the motion of the arm affects motion of the base.

SPART Toolbox

The Space Robotic Toolbox [25] is a package developed for kinematic and dynamic analysis of open-chain manipulators. The MATLAB-based toolbox accepts a Unified

[1]Model discrepancies here are perturbations in the mass and geometry of the dynamical system that the MPC routine is not made aware of.

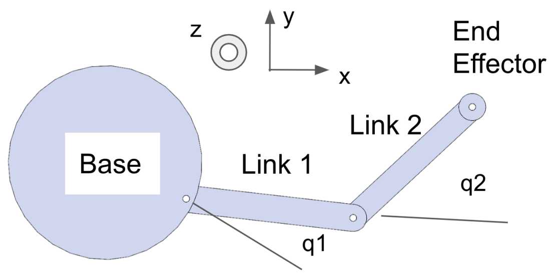


Figure 3.1: Planar model

Robot Description Format (URDF) representation of a manipulator system, and provides analysis tools for the provided model, allowing for both fixed-base and floating-base systems. The system equations of motion are computed using a recursive Newton-Euler algorithm (RNEA) method in the SPART toolbox.

Figure 3.2 depicts the formulation of state dynamics used in the study. Forward dynamics relate the current state and control inputs to the requisite state velocity. The applied control inputs are then combined to complete a forward dynamic calculation using RNEA to return joint and base accelerations. The forward dynamics are used for both trajectory optimization and simulation.

Planar Space Robot

In order to analyze the relevant operations presented in [21], a simplified planar dynamic model was created using a URDF and the SPART toolbox. We use a planar robot here to investigate the efficacy of MPC methods while easing the computational complexity of solving an optimization problem for a higher DoF, non-planar robot. Figure 3.1 and Table 3.1 describe the system used for planar analysis throughout the present study. The model consists of three bodies: a base and two links in an open-chain configuration. The two consecutive joints connecting the three bodies allow for rotational motion about parallel axes (axes parallel to the inertial z axis in Fig. 3.1), constraining the active motions to a plane. Each joint is active, providing a control vector of $u = [T_{j1}, T_{j2}]^T$. The base is free to float w.r.t. to the inertial frame, which introduces three additional coordinates for state description. This condition leads to an underactuated system, with minimum five generalized state coordinates, and two independent actuated inputs.

Parameter	value	unit
Base mass	12.9	kg
Base moment of inertia	0.208	kgm^2
Base center to joint 1	0.327	m
Link 1 mass	4.5	kg
Link 1 moment of inertia	0.32	kgm^2
Link 1 length	0.62	m
Link 2 mass	1.5	kg
Link 2 moment of inertia	0.049	kgm^2
Link 2 length	0.6	m

Table 3.1: Parameters for the major components of the planar model satellite

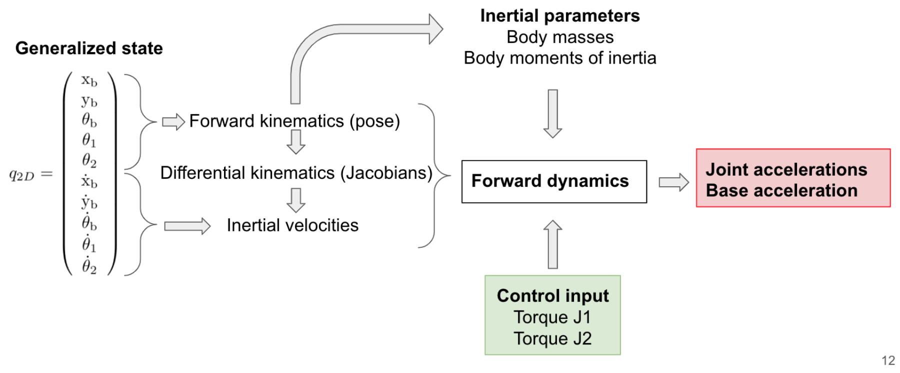

Generalized state
$q_{2D} = (x_b, y_b, \theta_b, \theta_1, \theta_2, \dot{x}_b, \dot{y}_b, \dot{\theta}_b, \dot{\theta}_1, \dot{\theta}_2)$
Forward kinematics (pose)
Differential kinematics (Jacobians)
Inertial velocities
Inertial parameters
Body masses
Body moments of inertia
Forward dynamics
Joint accelerations
Base acceleration
Control input
Torque J1
Torque J2
12

Figure 3.2: Dynamics flowchart

$$q_{Planar} \in \mathbb{R}^{10} : \tag{3.2}$$

$$q_{Planar} = [\mathrm{x_b}, \mathrm{y_b}, \theta_\mathrm{b}, \theta_1, \theta_2, \dot{x}_b, \dot{y}_b, \dot{\theta}_b, \dot{\theta}_1, \dot{\theta}_2] \tag{3.3}$$

Augmented State

Tasks required of the system generally relate to the end effector state. Since the generalized coordinates for describing the system dynamics do not directly provide end effector information, an augmented state with explicit end effector position and velocity in the inertial coordinate frame was created. The augmented state allows access to end effector state for trajectory optimization and control. The augmented state vector is shown in equation 3.4.

$q_{PlanarAug} \in \mathbb{R}^{14}$:

$$q_{PlanarAug} = \begin{pmatrix} \mathrm{x_b} \\ \mathrm{y_b} \\ \theta_\mathrm{b} \\ \theta_1 \\ \theta_2 \\ \dot{x}_b \\ \dot{y}_b \\ \dot{\theta}_b \\ \dot{\theta}_1 \\ \dot{\theta}_2 \\ \mathrm{EE_{posX}} \\ \mathrm{EE_{posY}} \\ \mathrm{EE_{velX}} \\ \mathrm{EE_{velY}} \end{pmatrix} \tag{3.4}$$

3.1.1 Space Robot Center of Mass Analysis

Before we discuss the Model Predictive Controllers we implemented on our planar space robot simulation, we fist verify that the planar space robot is obeying linear momentum conservation constraints. Because our robot is not firing any base thrusters and is not driving any reaction wheels, the total linear momentum of the system should stay fixed at zero. If we plot this out, we expect to see the position of the center of mass of the system remain at the initial position where it starts. Fig. 3.3 shows an open loop control sequence executed on our space robot. Fig. 3.4 show the plots of the X and Y positions of the system center of mass as a function of time. Now that we have shown that our system is properly obeying linear momentum constraints, we can feel confident that we have properly modeled a floating base system free from the effects of gravity.

3.1.2 Model Predictive Control

Model Predictive Control (MPC) is a control method that produces the optimal control action at each time step during operation through solving a local optimization

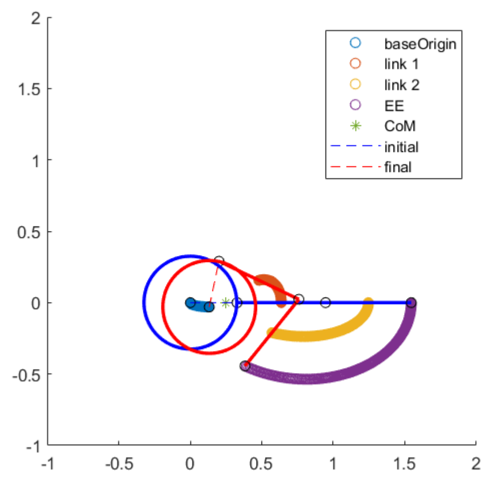


Figure 3.3: Open loop trajectory visualization. Note that the position of the system center of mass does not change.

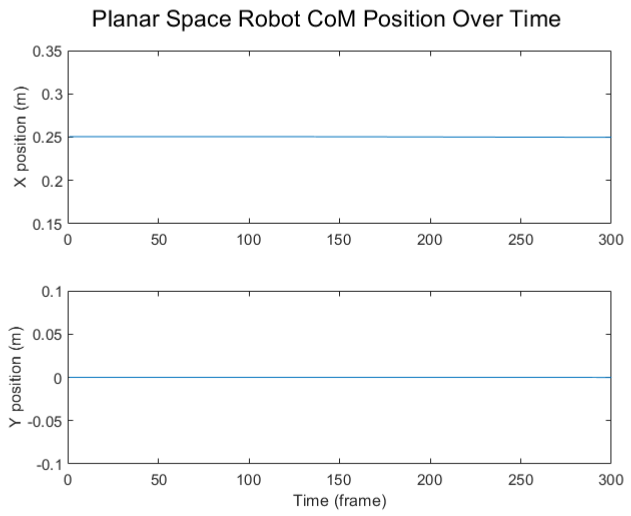


Figure 3.4: X-Y plots of planar space robot system center of mass over time. Note that the position of the system center of mass does not change over time. This is due to the fact that the robot is not firing any base thrusters, and is not experiencing any external contact forces.

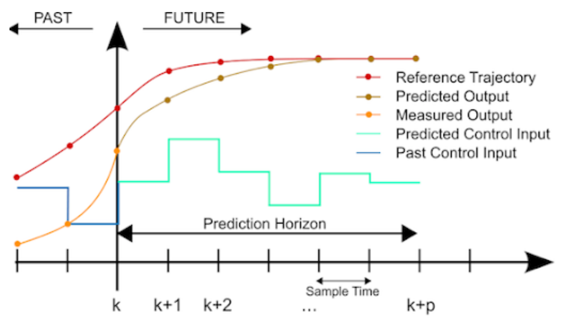


Figure 3.5: MPC graphical example. Image from [5].

problem [7] at that time step. Figure 3.5 depicts a temporal representation of the MPC process. The optimization is performed over a prediction horizon into the future, which shifts forward during operation as a ”receding horizon.” Different methods for model predictive control address trade-offs between optimization accuracy, horizon distance, cost definition, constraint definitions, and their effect on computational load. Model predictive controllers are model-based controllers, relying on the accuracy of the model relative to the representative system for efficacy. MPC methods differ from other optimal control techniques such as Linear Quadratic Regulation due to the run-time iterative process of optimization. MATLAB provides a set of tools for implementing model predictive controllers [17].

Adaptive MPC

Linear MPC methods require a linear time-invariant system for viable operation. The state dynamic equations exhibit configuration dependent relations, forming a significantly non-linear system over the space of configurations. Non-linear systems pose more complicated dynamics that in general cannot be analyzed directly with linear methods. Adaptive MPC addresses control of non-linear systems using linear MPC methods by applying linearizations of the dynamics at each time step. While continual linearizations add computational load to the system, the simplification of the optimization to a linear system significantly reduces the total decision load per time step. A linearization is only valid around an nominal trajectory of the system at which the approximation is applied. Thus for good performance, the relationship between the reference target and system state should be within reasonable proximity (trust region). Given a marginal state deviation over the time horizon to the reference value, a linear approximation of the non-linear dynamics can provide adequate dynamic accuracy to implement linear MPC methods on the linearized system.

In order to use linear MPC here, we must first linearize our system. Equations 3.5-3.7 describe the process for which we linearize our nonlinear dynamical system. [19]. Both the nominal state and control describe the linearization location, and the partial differentiation of the dynamics with respect to the state and control provides the first order description of the small deviation away from the local point.

$$\dot{\bar{x}} = f(\bar{x}, \bar{u}) \tag{3.5}$$

$$x(t) = \bar{x}(t) + \delta_x(t) \tag{3.6}$$

$$\dot{\bar{x}}(t) + \dot{\delta_x}(t) \approx f(\bar{x}, \bar{u}) + \frac{\partial f}{\partial x}|_{\bar{x},\bar{u}}\delta_x(t) + \frac{\partial f}{\partial u}|_{\bar{x},\bar{u}}\delta_u(t) \tag{3.7}$$

The MPC process can now perform optimization using the linear approximation from equation 3.7 as a replacement for the full non-linear model. In order to obtain the linearized system dynamics without an explicit analytical model, numerical differentiation can be performed using the method of finite differences. The MATLAB toolbox Adaptive Robust Numerical Differentiation [8] provides functions for generating the linearized dynamic equations with respect to state and control at an operating point, and is used for linearization in this work.

Non-linear MPC

When the posed control problem exhibits non-linear dynamics, non-linear or time varying constraints, and alternative cost functions that don't assume a globally convex form, the model predictive control process requires more sophisticated tools to perform the optimization. Generally an iterative optimization process is used to minimize the cost function, such as sequential quadratic programming (SQP), which serves as the default solving method for MATLAB NMPC toolbox. It is important to note the iterative optimization described here is iterated within the current time step to find the optimal control action, leading to substantial increase in computational demands for the controller. Fig. 3.6

Below is an overview of the sequential quadratic optimization method [17]:

1. Approximate optimization with a quadratic form

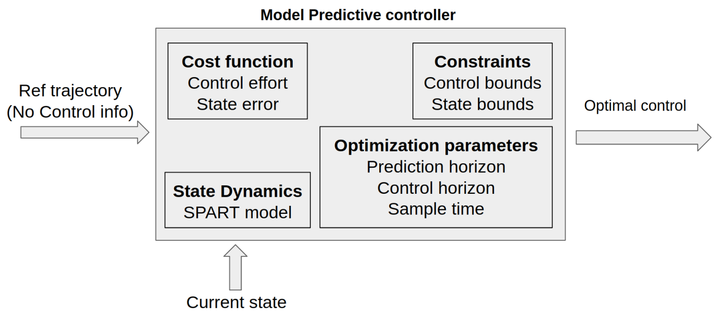


Figure 3.6: NMPC flowchart

2. Solve the Quadratic Program sub-problem
3. Iterate process at the new trajectory
4. Terminate process when a local minimum cost trajectory is found

3.1.3 Rendezvous Maneuver with MPC

The work in [21] based the realization of the planned trajectory around a non-linear MPC controller. Our work offers a comparison of adaptive MPC and non-linear MPC as related approaches to implementing the trajectory on the satellite model. Our control objective is to stabilize the position of the end-effector of the manipulator to a desired set point and its velocity to zero. The dynamic system described in section 3.1 with the augmented state was used to perform the target maneuver, and model predictive controllers were implemented using the MATLAB MPC toolbox. The controllers are passed the state trajectory without the nominal control sequence. This is deemed reasonable as the control is negligibly penalized in the cost function for the presented cases. Furthermore, the performance of the system following a desired trajectory without corresponding control information allows for a more general case of trajectory following. We do not pass in the nominal control sequence to better evaluate how the MPC methods handle the trajectory planning element of the control task. A further investigation would benefit from incorporation of the trajectory optimized control information to the control as a feed-forward term.

The maneuver was simulated with two different scenarios:

1. Coherence between the internal model parameters used for the optimization and actual system parameters (no differences)

2. Model discrepancies between component masses

In the latter scenario, the trajectories are planned about the model featuring discrepancies, the control synthesis algorithm computes the control inputs by performing an optimization, and the control sequences are applied to the actual, true system to iterate the state. Table 3.3 details the applied deviations between the models.

The outputs of the two different MPC methods applied to the rendezvous maneuver for both scenarios are summarized in Table 3.2. The performance of the controllers is compared with total run time and average error. Comparison with respect to run time shows Adaptive MPC outperforming the non-linear controller by an order of magnitude. This improvement is expected due the optimization benefits from linearization of the system dynamics, as detailed in section 3.1.2.

Fig	Scenario	Nominal Time	Run time	Avg error
3.10	NMPC - No model diff	4.0 s	493.10 s	0.0101 m
3.14	NMPC - Model diff	4.0 s	568.54 s	0.0126 m
3.18	Adaptive MPC - No model diff	4.0 s	63.41 s	0.0090 m
3.22	Adaptive MPC - Model diff	4.0 s	63.64 s	0.059 m

Table 3.2: Planar space robot trajectory tracking comparison - NMPC vs Adaptive MPC

Comparing output performance in end effector error is more nuanced. Given an alignment of model and actual system parameters, adaptive MPC and non-linear MPC both offer similar state performance. However, introducing model inaccuracies significantly degrades the performance of adaptive MPC as compared to NMPC, which demonstrates a higher degree of robustness to system parameter inaccuracies. Again this difference in performance is expected, as a linearization further introduces prediction error into the optimization process. The relationship between model inaccuracies and controller performance for the satellite system mirrors the results from the controllers applied to the simple pendulums. Given the significant model deviations and an abrupt shift in target motion near the end of the maneuver, the adaptive controller fails to provide the necessary adjustments to maintain target proximity.

This set of test conditions highlight particular trade-offs between adaptive and non-linear MPC. Extending the study would entail further refinement of cost functions, optimization of run time code to remove specific implementation discrepancies, and adjustment of other parameters such as the control and prediction horizons to investigate controller sensitivities.

Component	Model deviation
Base	+30% mass, Inertia
Link 1	-30% mass, Inertia
Link 2	-30% mass, Inertia

Table 3.3: Satellite component deviations from real to model

3.1.4 MPC Limitations

From our results, we have seen that Nonlinear MPC produces better trajectory tracking (even when there is an error in the formulation of the model), although this enhanced tracking comes at the cost of increased computational complexity. Adaptive MPC runs much faster, but suffers from poor trajectory tracking if the provided model is not accurate. Both Nonlinear MPC and adaptive MPC have key limitations in that they both do not run in real time, and in the case of adaptive MPC, are vulnerable to poor performance in the face of uncertainty in the model description. This is highlighted in Table 3.2, where we can see that neither method was capable of generating the four second trajectory in a timely manner. Because of these limitations, we decided to implement a classical control method (OSC) on our full 3D space robot simulation. One way to address these runtime limitations would be to implement these MPC methods using optimized C++ code, but we decided that implementing these methods in C++ would be out of the scope of this work.

[b]0.65

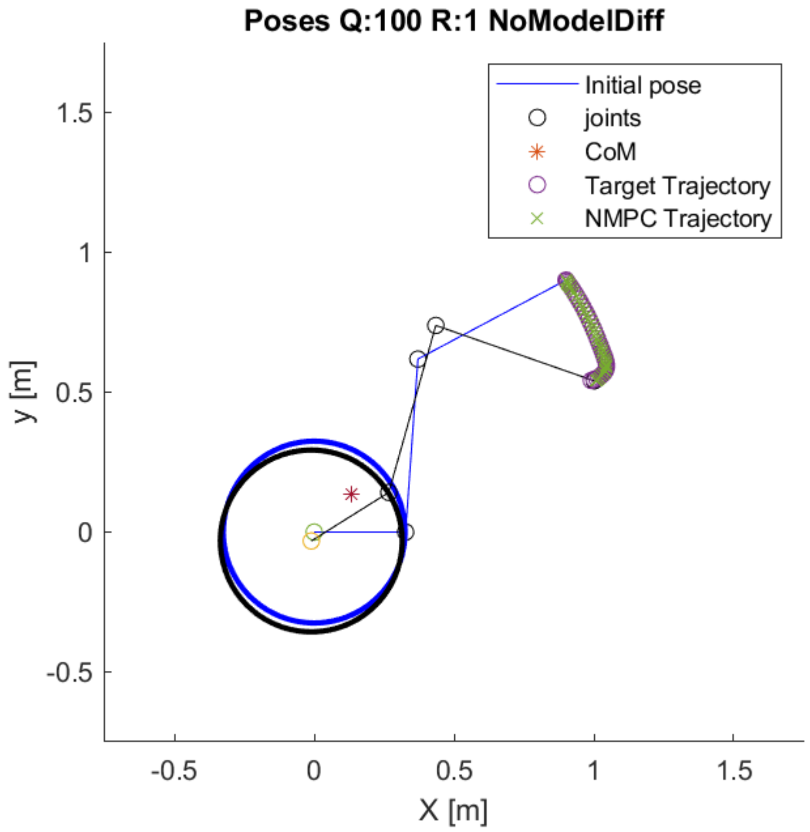


Figure 3.7: Initial and final pose

[b]0.65

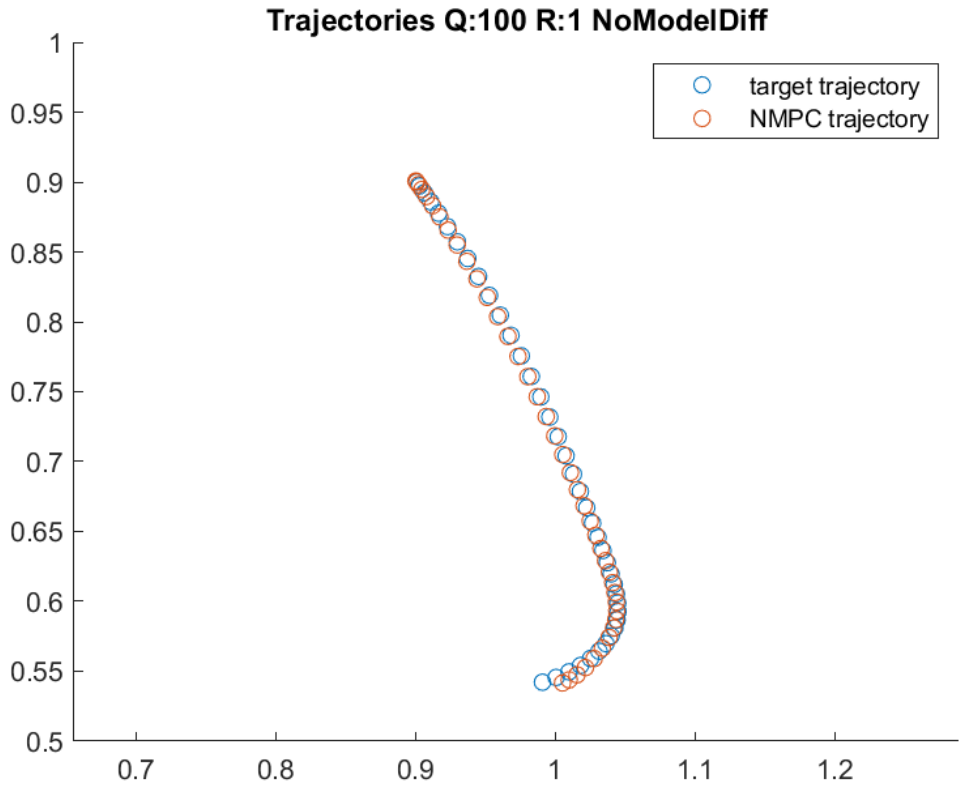


Figure 3.8: EE target vs realized

[b]0.65

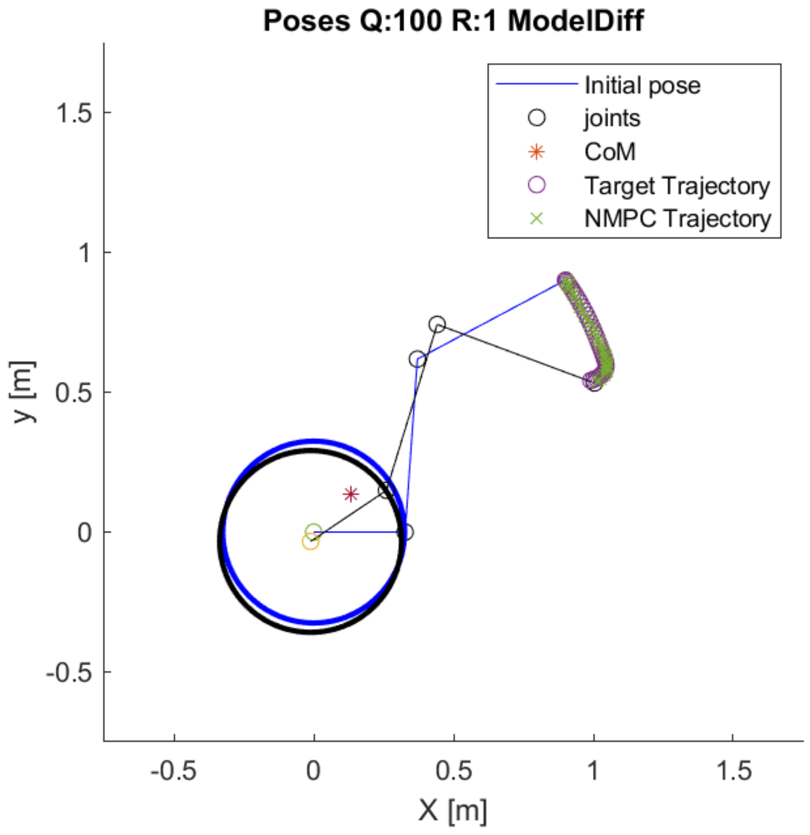


Figure 3.11: Initial and final pose

[b]0.65

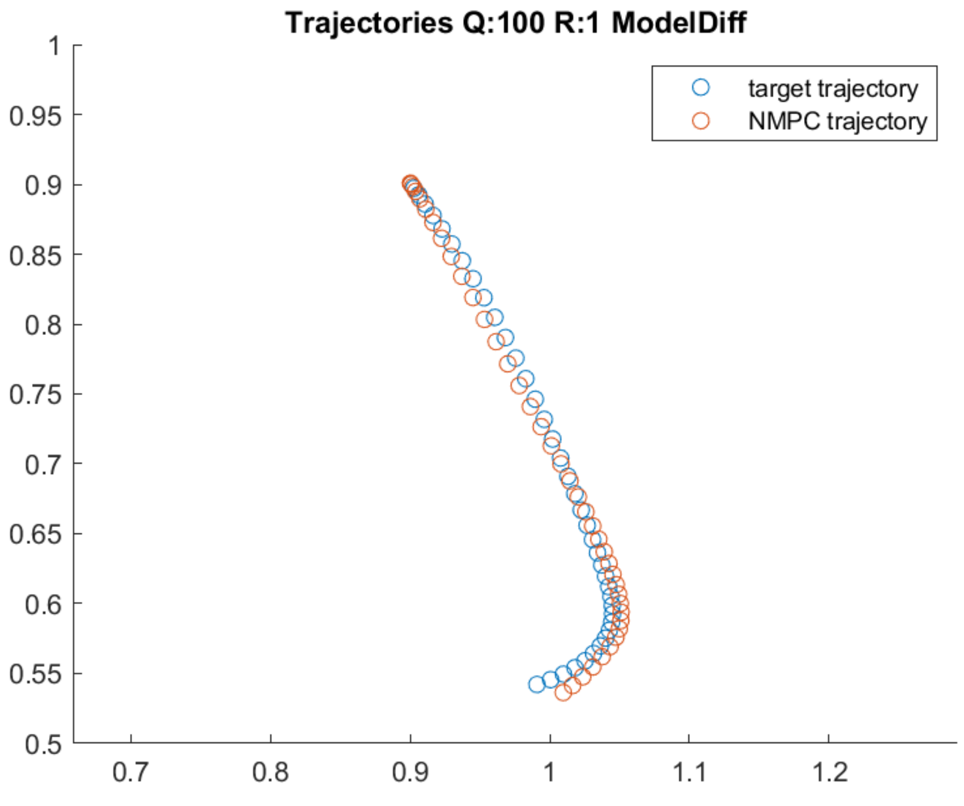


Figure 3.12: End effector target vs realized

[b]0.65

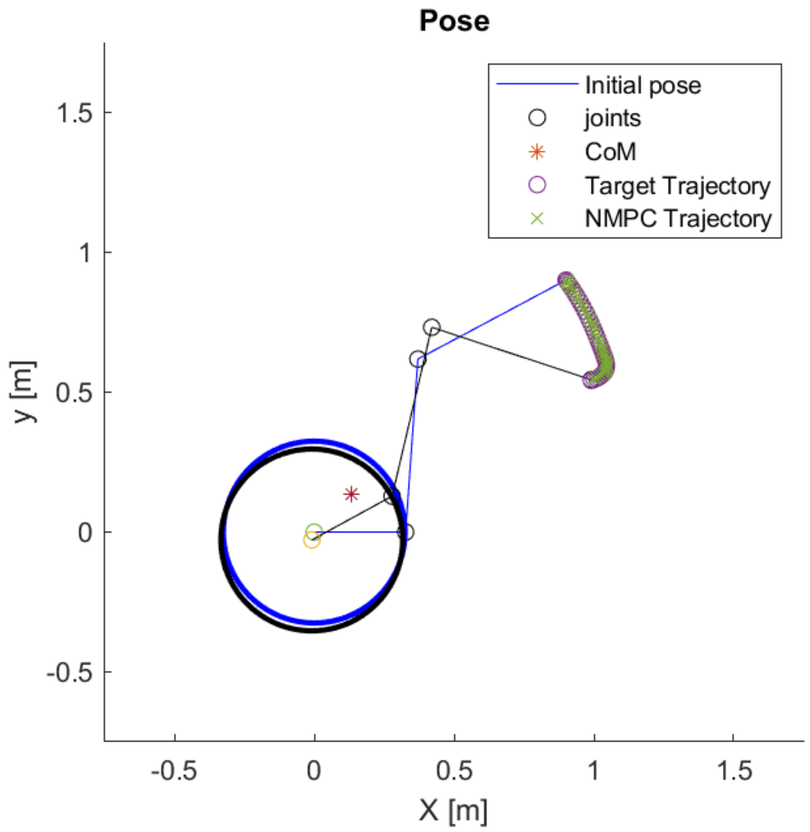


Figure 3.15: Initial and final pose

[b]0.65

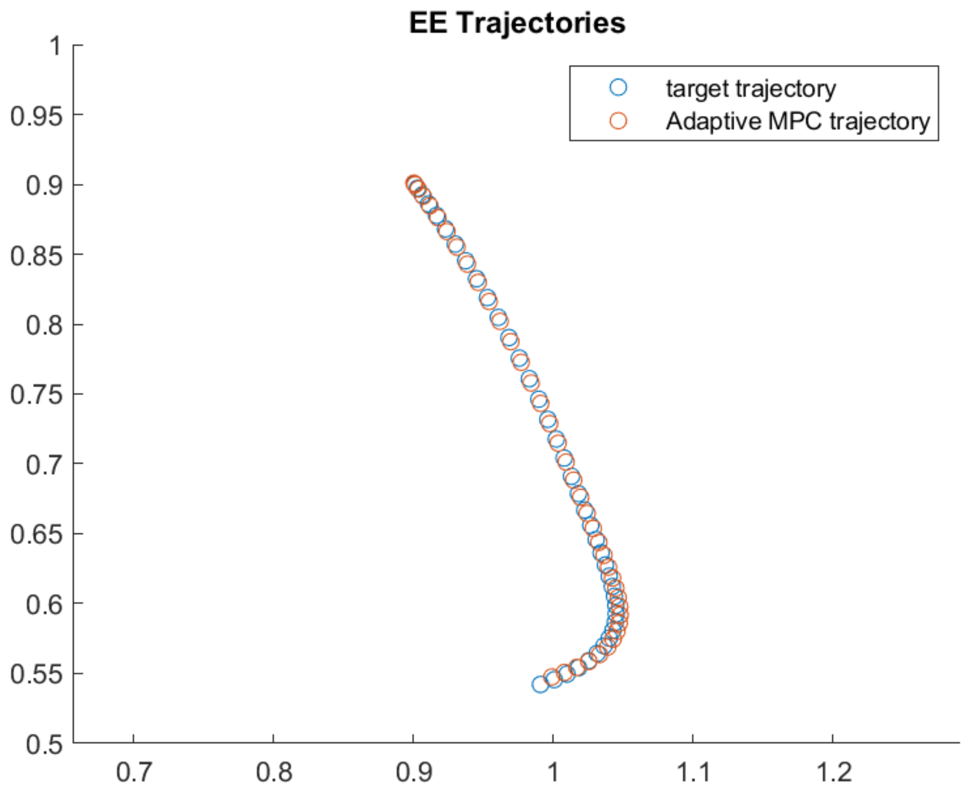


Figure 3.16: End effector target vs realized

[b]0.65

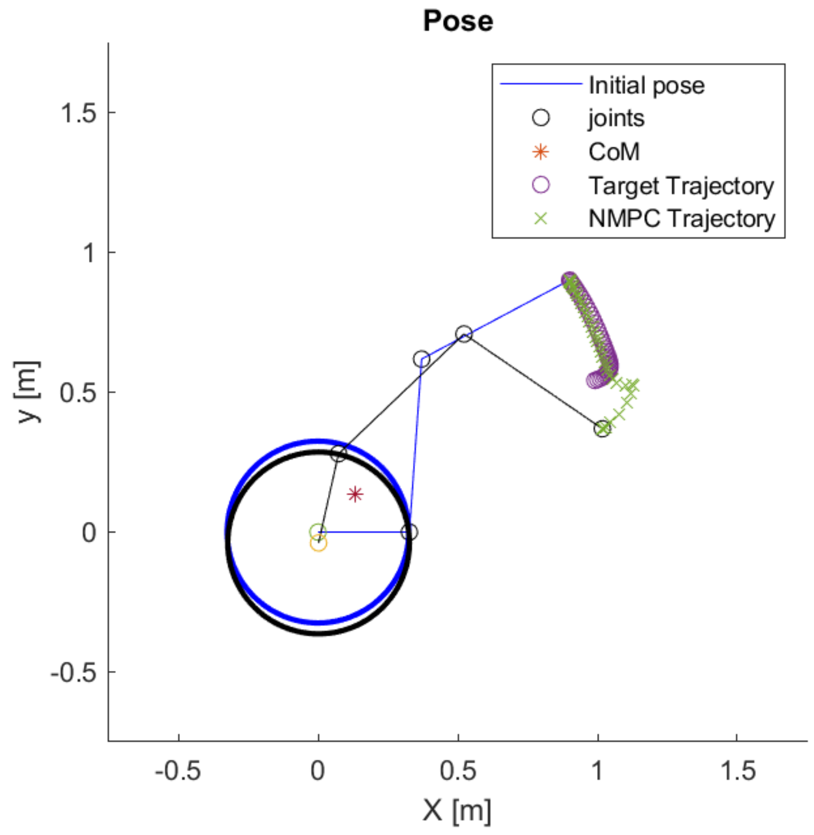


Figure 3.19: Initial and final pose

[b]0.65

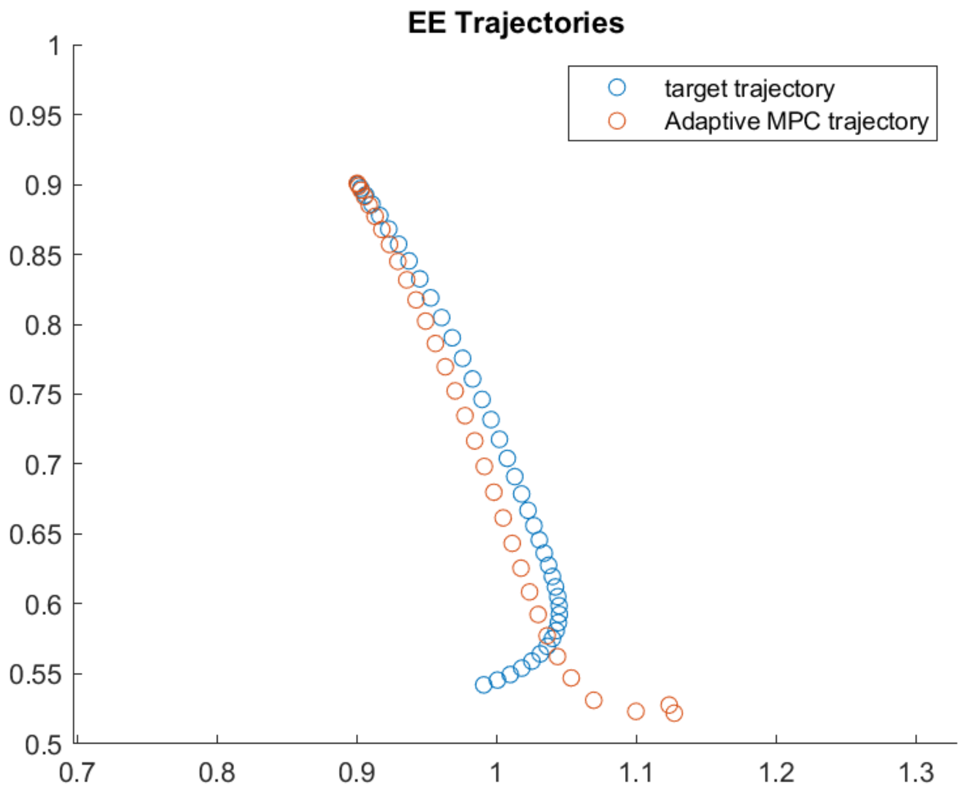


Figure 3.20: End effector target vs realized

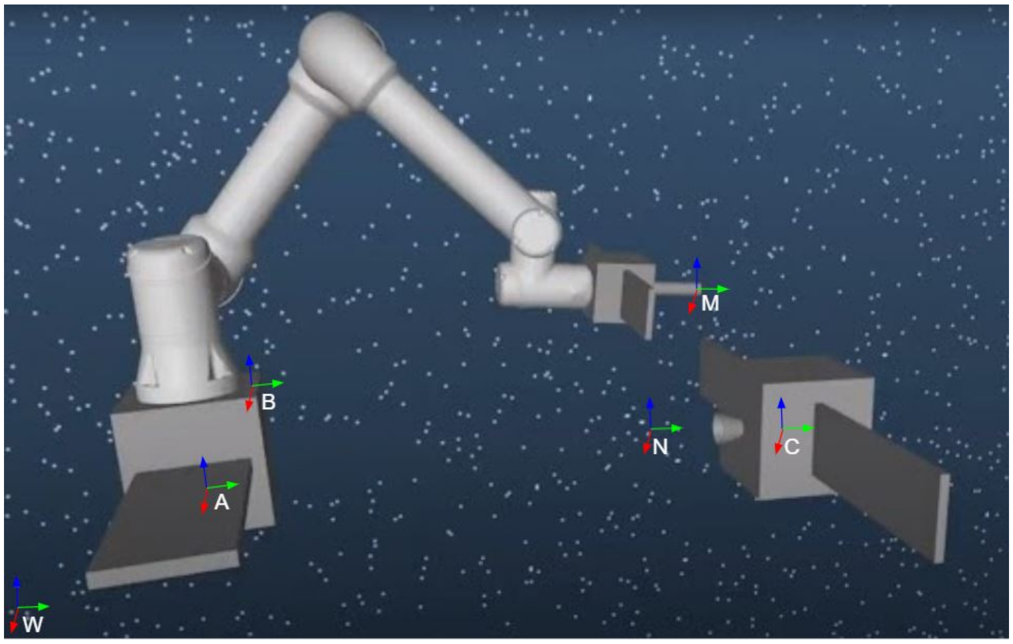


Figure 3.23: Relevant coordinate frames for OSC subsystem

3.2 Operational Space Control

Our OSC subsystem has two phases: approach and docking. During the approach phase, the base thrusters and reaction wheels are turned on until the MEP reaches the nominal point (coordinate frame N in Fig. 3.23). Once this point is reached, the OSC subsystem switches into the docking phase.

3.2.1 Controller Derivation - Approach Phase

Here we will briefly describe the controller that is active during the approach phase. The controller used in the approach phase is very similar to the controller used in the docking phase aside from adding forces/torques that drive the MRV base. In order to simplify this derivation, we assume that the reaction wheels to do not have any inertia, and can instantly impact torques onto the system, so the mass matrix and Coriolis terms remain the same. In reality, adding reaction wheels into the system dynamics would change the composition of the mass matrix and the Coriolis terms.

We start with the equations of motion for our system with an actuated base:

$$A_{sys} \begin{bmatrix} \ddot{x}_{base} \\ \ddot{q} \end{bmatrix} + b_{sys} = \begin{bmatrix} \Gamma_{base} \\ \Gamma_{arm} \end{bmatrix} + J^T F_{ext} \tag{3.8}$$

As we will describe in the derivation for the docking phase controller, these equations of motion can be transformed into the task space with the following form:

$$\Lambda \ddot{x} + \mu + p = J_{sys} \begin{bmatrix} \Gamma_{base} \\ \Gamma_{arm} \end{bmatrix} + F_{ext} \tag{3.9}$$

From here, we can derive our controller for the approach phase by inverting the J_{sys} matrix and pre-multiplying it onto both sides:

$$\begin{bmatrix} \Gamma_{base} \\ \Gamma_{arm} \end{bmatrix} = J_{sys}^{-1}(\Lambda \ddot{x}_d + \mu + p - F_{ext}) \tag{3.10}$$

3.2.2 Controller Derivation - Docking Phase

The following is inspired by the work done by Sentis et al. on operational space control for legged robots [22]. We first start with the full system equations of motion obtained from section 2.2:

$$A_{sys} \begin{bmatrix} \ddot{x}_{base} \\ \ddot{q} \end{bmatrix} + b_{sys} + g_{sys} = \begin{bmatrix} 0 \\ \Gamma \end{bmatrix} + J^T F_{ext} \tag{3.11}$$

Where $\ddot{x}_{base} \in \mathbb{R}^{6\times1}$ is the vector of base accelerations, $\ddot{q} \in \mathbb{R}^{6\times1}$ is the vector of manipulator arm joint accelerations, $b_{sys} \in \mathbb{R}^{12\times1}$ is the vector containing Coriolis terms, $F_{ext} \in \mathbb{R}^{6\times1}$ in the external wrench applied to the end effector of the space robot. $g_{sys} \in \mathbb{R}^{12\times1}$ represents gravitational terms, which for our space system can be neglected. So we have (for the docking phase):

$$A_{sys} \begin{bmatrix} \ddot{x}_{base} \\ \ddot{q} \end{bmatrix} + b_{sys} = \begin{bmatrix} 0 \\ \Gamma \end{bmatrix} + J^T F_{ext} \tag{3.12}$$

Where J is given by

$$\begin{bmatrix} J_{base} & J_{arm} \end{bmatrix} \tag{3.13}$$

We can break down $A_{sys} \in \mathbb{R}^{12\times12}$ into four block matrices:

$$A_{sys} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \tag{3.14}$$

Now, we can write out equation 3.12 in a system of two equations that separate the dynamics of the base and of the arm:

$$A_{11}\ddot{x}_{base} + A_{12}\ddot{q} + b_{sys}^{base} = 0 + J_{base}^T F_{ext} \tag{3.15}$$

$$A_{21}\ddot{x}_{base} + A_{22}\ddot{q} + b_{sys}^{arm} = \Gamma + J_{arm}^T F_{ext} \tag{3.16}$$

We can replace $\ddot{x}_{base}$ in equation 3.16 by solving for $\ddot{x}_{base}$ in equation 3.15:

$$A_{21}A_{11}^{-1}(-A_{12}\ddot{q} - b_{sys}^{base} + J_{base}^T F_{ext}) + A_{22}\ddot{q} + b_{sys}^{arm} = \Gamma + J_{arm}^T F_{ext} \tag{3.17}$$

Now, let's rewrite the above equation and group terms:

$$(A_{22} - A_{21}A_{11}^{-1}A_{12})\ddot{q} + (b_{sys}^{arm} - A_{12}A_{11}^{-1}b_{sys}^{base}) \tag{3.18}$$
$$= \Gamma + (J_{arm}^T - A_{21}A_{11}^{-1}J_{base}^T)F_{ext} \tag{3.19}$$

Now we can define the following matrices for convenience:

$$A^{'} = (A_{22} - A_{21}A_{11}^{-1}A_{12}) \tag{3.20}$$

$$b^{'} = (b_{sys}^{arm} - A_{12}A_{11}^{-1}b_{sys}^{base}) \tag{3.21}$$

$$J^{*T} = (J_{arm}^T - A_{21}A_{11}^{-1}J_{base}^T) \tag{3.22}$$

$$J^{*} = (J_{arm} - A_{21}A_{11}^{-1}J_{base}) \tag{3.23}$$

Now, let's write out our simplified equations of motion in the joint space:

$$A^{'}\ddot{q} + b^{'} = \Gamma + J^{*}F_{ext} \tag{3.24}$$

Now, we want to control behavior in the task space, so we can transform these actuated joint space dynamics into the task space by pre-multiplying both sides with the transpose of $\bar{J}$, the dynamically consistent inverse of J, where $\bar{J} = A^{'-1}J^{*T}\Lambda$:

$$\bar{J}^T(A^{'}\ddot{q} + b^{'}) = \bar{J}^T(\Gamma + J^{*}F_{ext}) \tag{3.25}$$

This yields our task space dynamics:

$$\Lambda\ddot{x} + \mu = F + F_{ext} \tag{3.26}$$

Where $\Lambda = (J^* A'^{-1} J^{*T})^{-1}$ is the OSC mass matrix, and μ is the OSC Coriolis vector.

Now, we must design a controller, F, to cancel out the Coriolis and inertia terms in our task space dynamics:

$$F = \bar{J}^T J^{*T}(\Lambda\ddot{x}_d + \mu - F_{ext}) \tag{3.27}$$

Where $\ddot{x_d}$ is the desired task space acceleration, given by our ideal spring-mass damper equations:

$$\ddot{x}_d = -k_p(x - x_d) - k_v\dot{x} \tag{3.28}$$

When we substitute our controller into our task space dynamics, we get the following:

$$\Lambda\ddot{x} + \mu = \bar{J}^T J^{*T}(\Lambda\ddot{x}_d + \mu - F_{ext}) + F_{ext} \tag{3.29}$$

Where $\bar{J}^T J^{*T}$ is cancelled out, because $\bar{J}$ is the generalized inverse of J^*, thus leaving us with:

$$\Lambda\ddot{x} + \mu = \Lambda\ddot{x}_d + \mu - F_{ext} + F_{ext} \tag{3.30}$$

This leaves us with:

$$\ddot{x} = \ddot{x}_d \tag{3.31}$$

Or,

$$\ddot{x} = -k_p(x - x_d) - k_v\dot{x} \tag{3.32}$$

Equation 3.27 gives our operational space control generalized force. This generalized force is computed at each time step during simulations to bring the MEP to its desired location. Equation 3.32 shows that the resulting accelerations in the workspace will be equivalent to that of a spring mass damper connecting the MEP to the desired goal location.

3.2.3 Singularity Analysis of Generalized Jacobian

We defined the generalized Jacobian for our OSC subsystem as J^*, above. Here, we observe what MRV configurations result in dynamic singularities of the generalized jacobian. If a dynamic singularity were to occur (for example, when the MRV arm is fully outstretched), our OSC controller will not be able to function, as the generalized jacobian will no longer be invertible, and we will lose the ability to command sensible joint torques.

To evaluate the presence of dynamic singularities in our configuration space, we sampled 15,625 different initial joint configurations of the MRV arm while keeping the MRV base fixed. We found that of those 15,625 joint configurations, 10,000 of those configurations represent non-singular configurations. Roughly one third of the sampled joint configurations represented dynamic singularities, with the majority of these configurations being associated with arm links that are nearly aligned or fully outstretched. We sampled uniformly between plus and minus 45 degrees from the MRV arm's zeroed out pose (all joint angles set to zero). Table 3.4 summarizes these results. Figs. 3.24 and 3.25 show course representations of areas where dynamic singularities are present in the configuration space.

3.2.4 Center of Mass Position Analysis for Spatial MRV System

Before we discuss the feasibility analysis or the stability analysis performed for the MRV, we want to show how the base thrusters and reaction wheels influence the

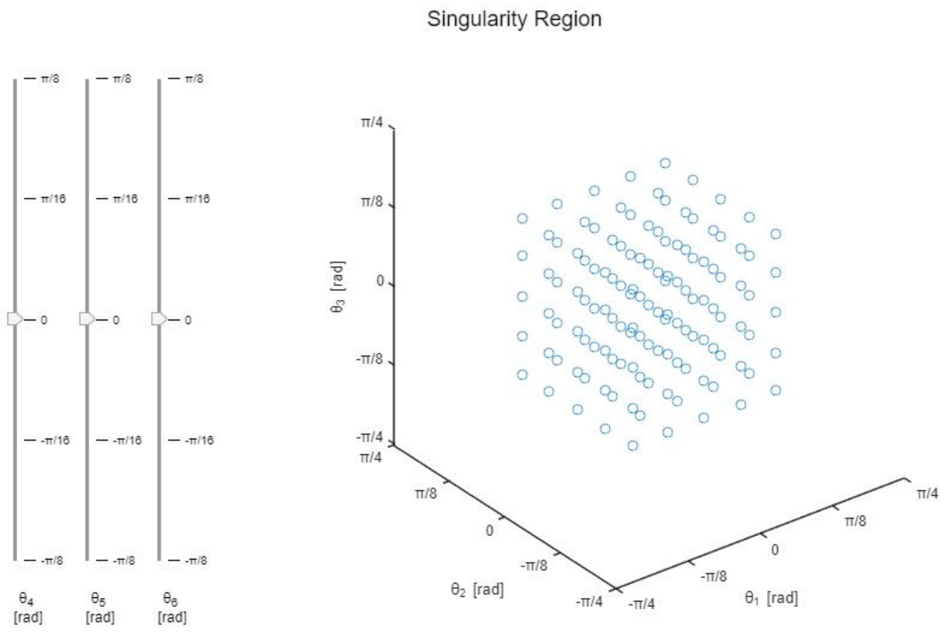


Figure 3.24: Scatter plot of region containing dynamic singularities. Note: $\theta_4, \theta_5, \theta_6$ are all set to zero to reduce the dimensionality of this visualization down to 3 from 6.

position of the center of mass during a simulated docking mission. Fig. 3.26 show time histories of the center of mass position of the MRV system during a simulated docking mission. We see small oscillations once the system enters the docking phase due to numerical imprecision in mujoco. Because we see the position of the system center of mass stop changing once the MRV base thrusters are turned off during docking, we can have additional confidence that we have properly modeled the floating base MRV system.

3.2.5 Analysis Overview

Now we discuss the analysis performed for our operational space controller. The analysis we performed can be grouped into two categories: Stability analysis and feasibility analysis. Feasibility analysis consists of determining what is the set of possible states (or initial conditions) for which the control system is guaranteed to succeed in the insertion task. Stability analysis involves determining how to characterize the global behavior of the controller, and to provide theoretical guarantees

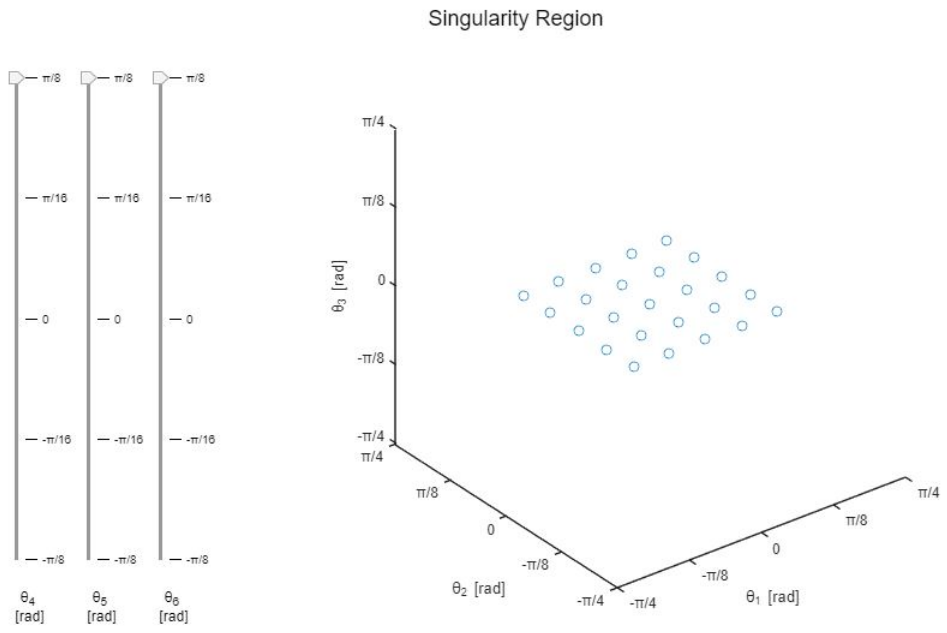


Figure 3.25: Scatter plot of region containing dynamic singularities. Note: $\theta_4, \theta_5, \theta_6$ are all set to $-\pi/16, -\pi/8$, and $\pi/16$ to reduce the dimensionality of this visualization down to 3 from 6.

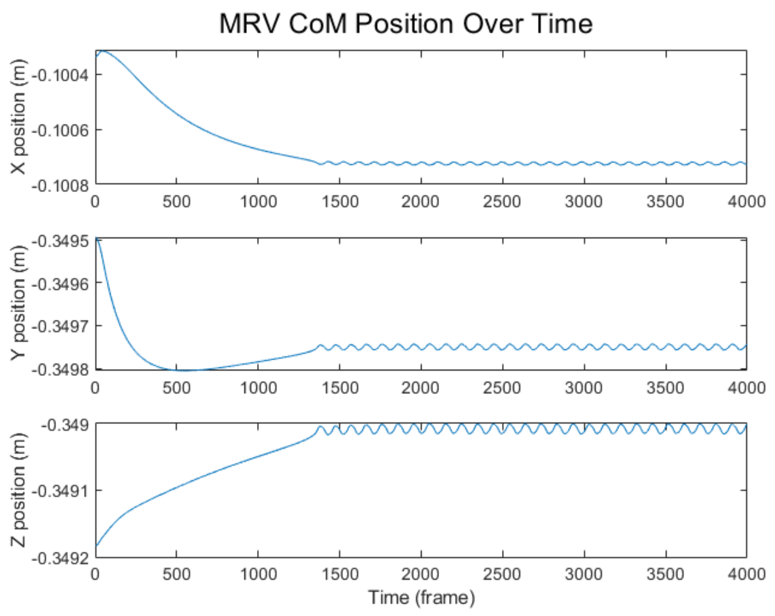


Figure 3.26: MRV center of mass position plotted over time. Note at frame 1378 how the position of the system center of mass stops changing. This is the moment at which the OSC controller shifts from the approach phase to the docking phase, and the MRV base thrusters and reaction wheels are turned off.

MRV Dynamic Singularities				
Sampled joint angle lower bound (rad)	Sampled joint angle upper bound (rad)	# of samples taken	# of singularities found	% singular configuration
-0.38	0.38	15625	5625	0.36

Table 3.4: Summary table for analysis of MRV dynamic singularities.

that the control can command generalized forces that bring the steady state error to zero.

3.2.6 Stability Analysis (Linear Control Theory)

Below is our derivation for proving stability for the full floating system under contact (note that here we avoid using the selection matrix by instead leveraging the coupled nature of the system dynamics and making substitutions between base and joint accelerations):

From equation 3.32, we now have our idealized spring-mass damper dynamics in the workspace.

We can convert these dynamics into state space representation using the following notation:

$$\bar{x} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \tag{3.33}$$

$$\bar{x} = \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} \tag{3.34}$$

Using equations 3.33 and 3.34, as well as 3.32, we can group terms into the following linear relationship:

$$\dot{\bar{x}} = A\bar{x} \tag{3.35}$$

Where A in this case is defined as $A = \begin{bmatrix} -k_p & 0 \\ 0 & -k_v \end{bmatrix}$

Now, if we perform eigenvalue analysis on linear A matrix, we find that our proposed controller is stable $\forall k_p, k_v > 0$

3.2.7 Stability Analysis (Lyapunov Theory)

Now, our goal is to reach some desired joint-space configuration, q_d, with a desired joint velocity, $\dot{q} = 0$, and desired joint acceleration, $\ddot{q} = 0$. Then we need to design a controller, Γ to bring $q, \dot{q}, \ddot{q}$ to these values.

Let us design the following Lyapunov candidate:

$$V = \frac{1}{2}(q - q_d)^T(q - q_d) + \frac{1}{2}\dot{q}^T\dot{q} \tag{3.36}$$

And the time derivative of the Lyapunov function:

$$\dot{V} = (q - q_d)^T(\dot{q} - \dot{q}_d) + \dot{q}^T\ddot{q} \tag{3.37}$$

We can substitute in the dynamics from equation 3.24 into the derivative of our Lyapunov candidate:

$$\dot{V} = (q - q_d)^T\dot{q} + \dot{q}^T[A^{'-1}(-b^{'} + \Gamma + J^{*T}F_{ext})] \tag{3.38}$$

We can design a suitable controller as follows:

$$\Gamma = b^{'} - J^{*T}F_{ext} - A^{'}(q - q_d) - A^{'}\dot{q} \tag{3.39}$$

If we substitute our controller into equation 3.37, we get:

$$\dot{V} = (q - q_d)^T\dot{q} + \dot{q}^T[A^{'-1}(-b^{'} + b^{'} - J^{*T}F_{ext} - A^{'}(q - q_d) - A^{'}\dot{q} + J^{*T}F_{ext})] \tag{3.40}$$

This first set of cancellations leaves us with:

$$\dot{V} = (q - q_d)^T \dot{q} + \dot{q}^T [A'^{-1}(-A'(q - q_d) - A'\dot{q})] \tag{3.41}$$

This can be simplified down to the following:

$$\dot{V} = -\dot{q}^T \dot{q} \tag{3.42}$$

Which is negative semi-definite. Therefore our control system is marginally stable.

3.2.8 Feasibility Analysis

We perform several variants of feasibility analysis to better understand the region of attraction of our operational space controller. In particular, we numerically evaluate the basins of attraction of two operational space controllers: The blind OSC, which only has access to the goal state in the Z axis along with force feedback in six DoF, and the fully observable OSC, which has access to the full goal state (x,y, and z positions) along with force feedback in six DoF. For both of these situations, we disconnect our ViSP tracker and state estimation pipeline from the controller, and provide the operational space controller with the ground truth goal locations. We disconnect our perception pipelines because we want to solely analyze the performance of our operational space controller, without influencing results by introducing the noise inherent in our perception pipeline.

We break our feasibility analysis into these two parts so we can evaluate two separate components of the controller. We want to be able to understand how the force feedback component of our controller reacts to external contact forces. First, we start by studying the feasibility of our blind OSC. These blind insertion tests involve starting the MEP at different positions and orientations in front of the client vehicle, and moving the MEP forward in the Z axis. When a collision occurs, we can observe whether or not the force feedback component of our operational space controller can account for these forces and still successfully insert into the client vehicle's rocket nozzle. Fig. 3.27 shows an pictorial overview of our feasibility experiments.

We perform a fully observable version of the feasibility study in order to verify the global stability claims we make in the stability analysis section above. By giving our operational space controller access to the full goal state without any noise, we can search for the full set of initial states for which our controller can drive the MEP into the client vehicle's rocket nozzle. Fig. 3.28 shows an example of the basin of attraction in three dimensions when the pitch and yaw angles are fixed at zero degrees.

3.2.9 Blind Insertion Results

For the blind insertion feasibility tests, we find that the region of success is a small region (which is a subset of the fully observable success region) that is nearby the radius of the client vehicle nozzle. See Table 3.5 for detailed bounds on the basin of attraction.

Blind Insertion Success Region Bounds						
X bounds (cm)	Y bounds (cm)	Z bounds (cm)	Pitch bounds (deg)	Yaw bounds (deg)	Success (%)	XYZ basin volume (cm^3)
±3.0	±3.0	±10.0	±6.0	±6.0	38	720
±1.0	±2.0	±10.0	±2.0	±2.0	80	160
±1.0	±1.0	±10.0	±2.0	±2.0	100	80

Table 3.5: Boundary of region of attraction for blind insertion tests.

Examining the blind insertion results, we can see that as we increase the value of the percentage of successful[2] trials (Table 3.5 column 6), the size of the XYZ region volume decreases. The XYZ volume is an intuitive way to represent the size of the region of attraction of our five dimensional search space. We choose this XYZ volume representation as a measure of the size of our region of attraction because it is difficult to describe the volume of a higher dimensional space which is measured with different units in different dimensions (meters in XYZ, degrees/radians in roll,pitch,yaw). From these blind insertion results we can see that even if there is some error in the positioning of the MEP (i.e. the MEP is off of nozzle centerline), we can still obtain

[2]We define success here as a trial which ends with the MEP within five millimeters of the prescribed goal position inside the client vehicle rocket nozzle. A failure occurs when the MEP fails to reach the goal position after two minutes elapse in simulation time.

high probabilities of a successful insertion. Without the use of the force feedback component within our controller, this would not be possible.

3.2.10 Fully Observable Insertion Results

We visualize the region of attraction we computed in our fully observable insertion tests in Figs. 3.28 and 3.29. We can see that this region is quite large in areas that have a small offset distance in the Z dimension from the client vehicle rocket nozzle. Qualitatively, this tells us that our controller is quite effective at inserting the MEP into the client vehicle's rocket nozzle. One way to interpret these basin of attraction figures is to look inside the red volume, and understand that any configuration of the MEP (at zero roll, pitch, and yaw angles) within this red region is guaranteed to result in a successful insertion. To find this region of attraction, we sampled 15,625 different initial conditions of the MEP's position relative to the origin defined in Fig. 3.27. Table 3.6 gives information about the space of initial conditions that was sampled in order to compute the region of attraction. We selected these search bounds due to the constraint that the floating base manipulator arm workspace imposes onto our system. Without base thrusters active, there is a limited accessible volume to sample initial conditions from.

Fully Observable Insertion Success Region Bounds				
X bounds (m)	Y bounds (m)	Z bounds (m)	Pitch bounds (deg)	Yaw bounds (deg)
± 0.4	± 0.4	± 0.1	± 0.0	± 0.0

Table 3.6: Boundary of sampled space for fully observable insertion tests. (Note that yaw and pitch were zeroed out in order to reduce the search space from a five dimensional space down to a three dimensional space).

3.2.11 Limitations

Our approach relies on a crucial assumption: we must have perfect knowledge of the kinematic and dynamic properties of the system. If the model we use to perform inverse dynamics does not match the true system dynamics, then our inverse dynamics will not cancel out the nonlinear task space dynamics as intended, and our controller will fail. This can be seen in Figs. 3.30 and 3.31. In Fig. 3.30, we see the error converges to zero as the model used by our OSC subsystem is unaltered. In Fig. 3.31, we see the error diverges, as we altered the masses of all of the links by doubling them.

Because our method relies on having perfectly modeled dynamics, we find that our OSC approach is primarily well-suited for applications in simulation where the true system model is known. For applications on real robots or in simulations where the true system dynamics are not perfectly known, an optimization-based method may be more appropriate to help accommodate for uncertainties in the true system dynamics. We did not implement any optimization based methods in an optimized language such as C++ because we decided that implementing such methods in C++ was outside the scope of this work.

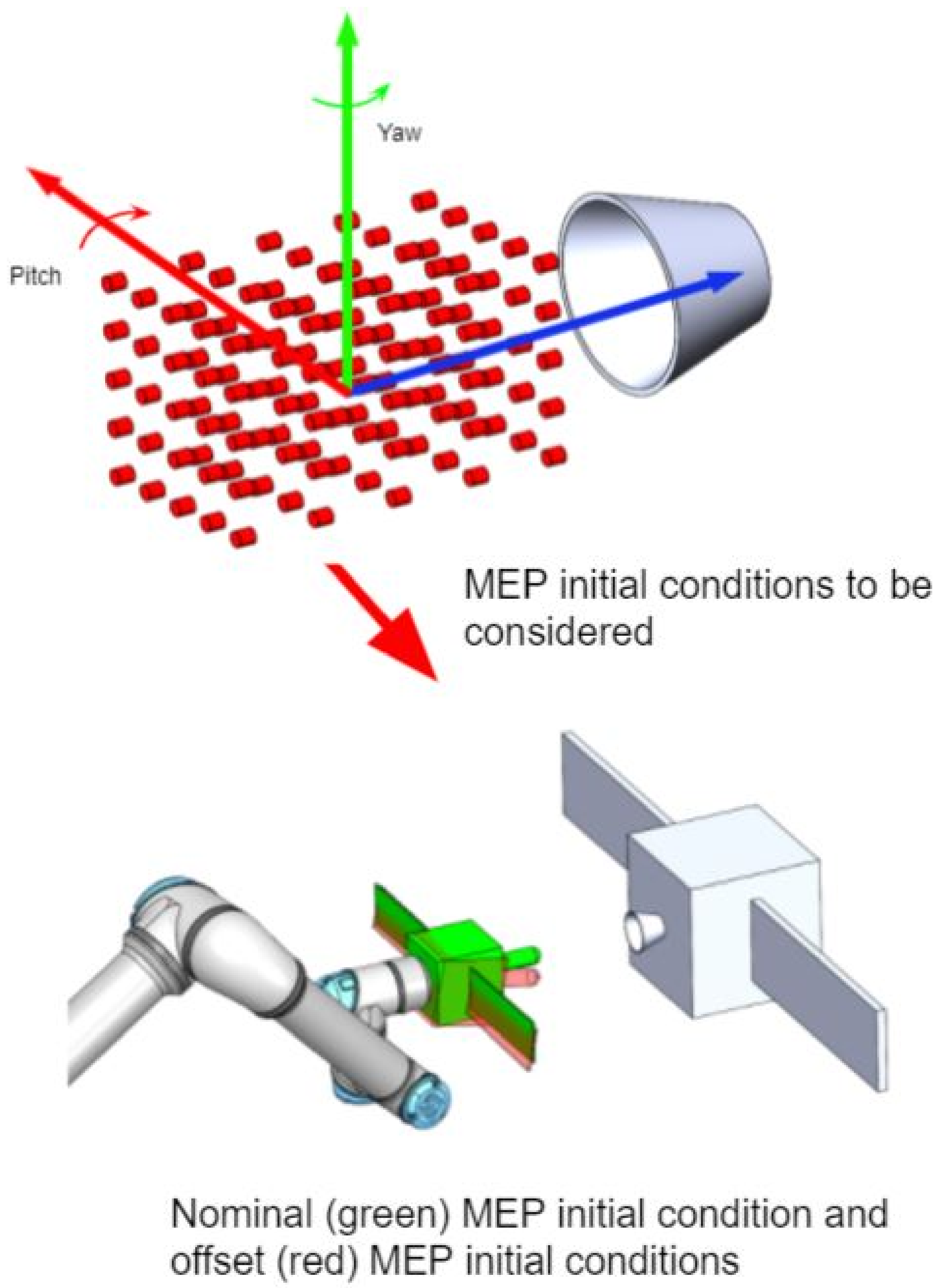

Yaw
Pitch
MEP initial conditions to be considered
Nominal (green) MEP initial condition and offset (red) MEP initial conditions

Figure 3.27: Overview of feasibility experiments. We sample a series of initial conditions outside of the client vehicle's rocket nozzle, and observe the performance of our operational space controller at each initial condition. The green shaded MEP represents the MEP being positioned at the origin of the point cloud of red cylinders above. The red shaded MEP represents a sampled location of the origin, but with a small rotation about the pitch axis.

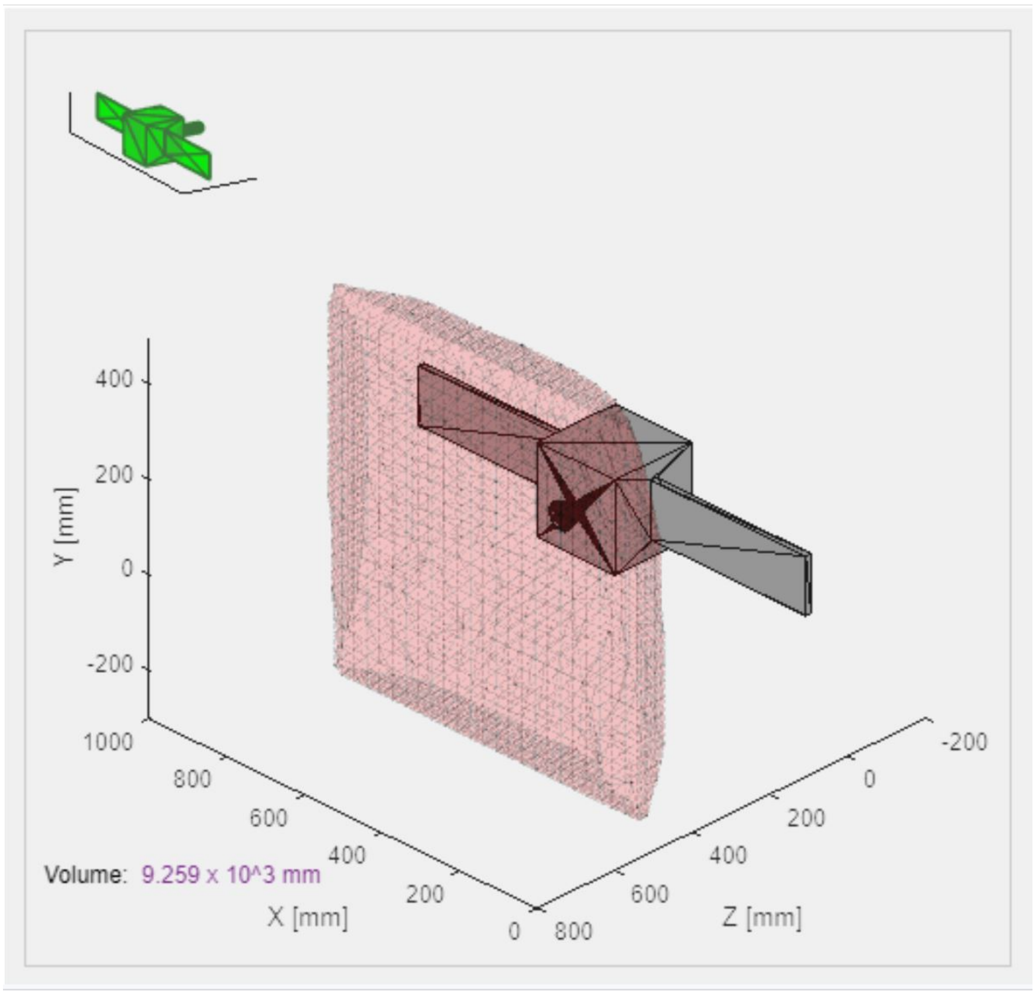


Figure 3.28: Example basin of attraction for fully observable feasibility experiment. Pitch=Yaw=0 degrees.

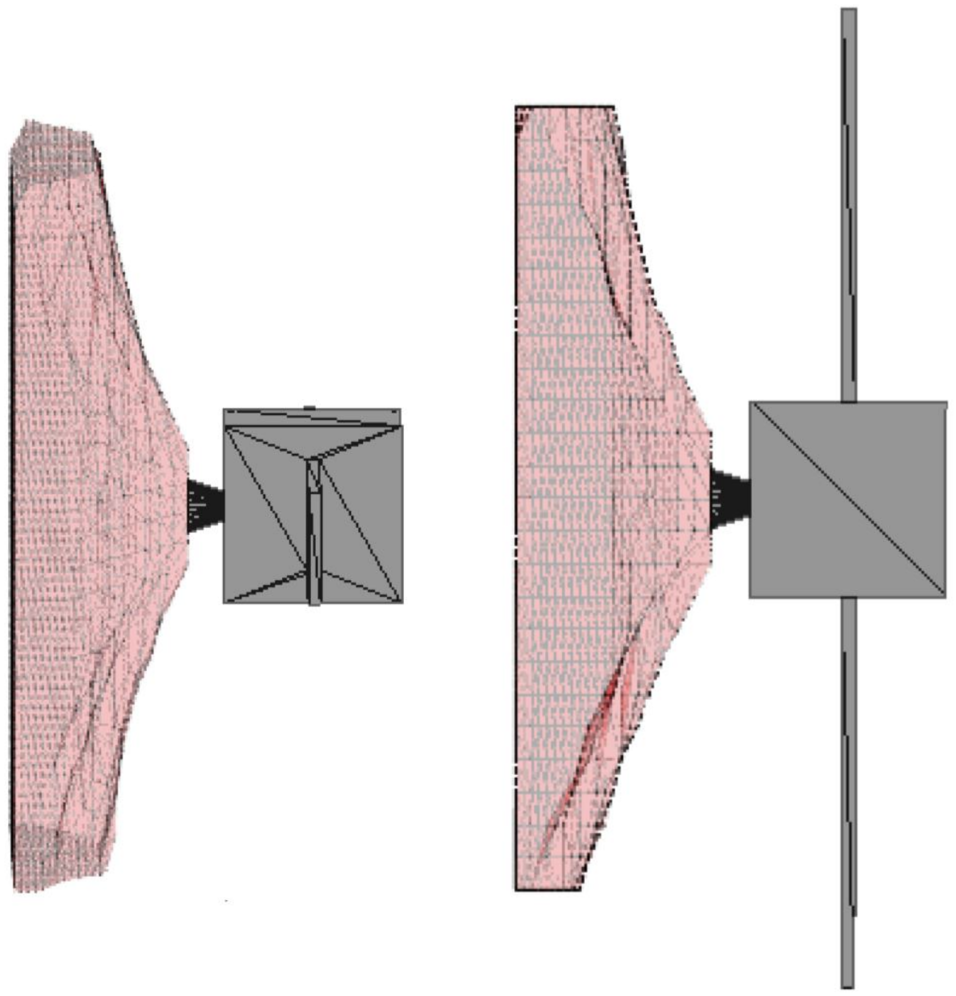


Figure 3.29: Example basin of attraction for fully observable feasibility experiment. The MEP orientation is set to Roll=Pitch=Yaw=0 degrees. Left: A side view of the region. Right: A top view of the region

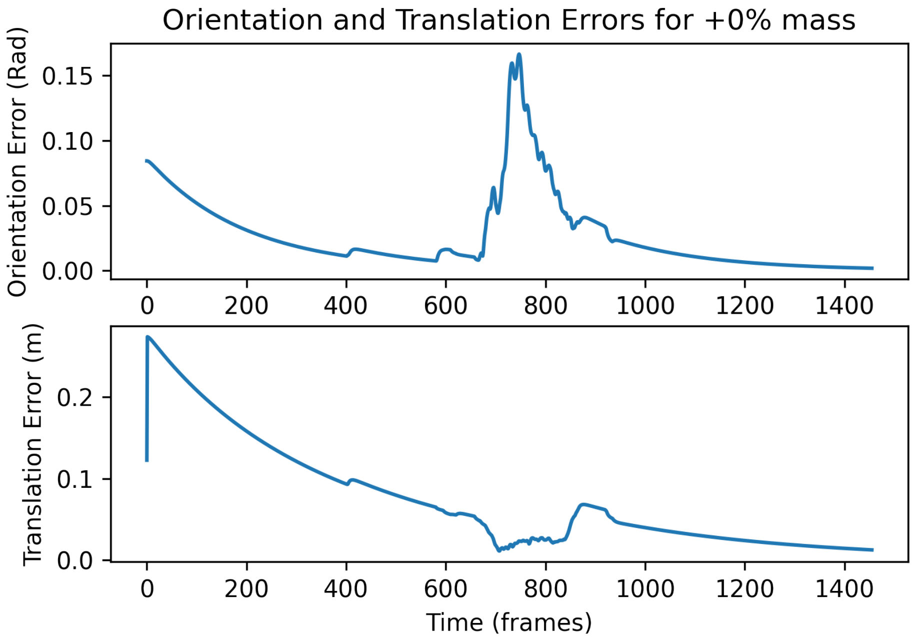


Figure 3.30: Error plots during docking simulation with true system masses passed into OSC

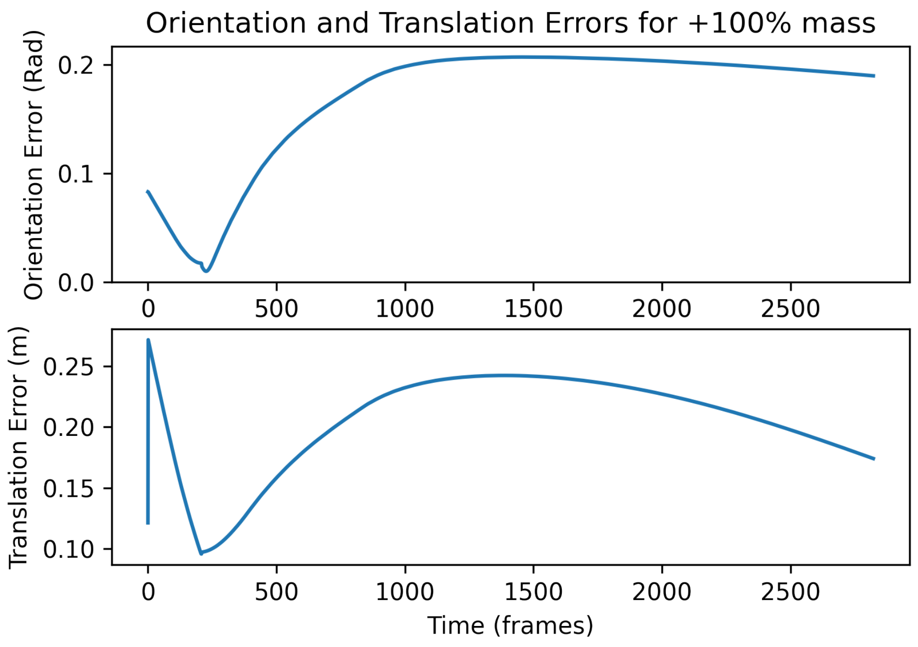


Figure 3.31: Error plots during docking simulation with altered system masses passed into OSC

Chapter 4

State Estimation for Space Robot

4.1 Tracking

4.1.1 ViSP Overview

In order to obtain pose measurements of the client vehicle, we leverage an existing tracking solution from INRIA called the Visual Servoing Platform, or ViSP [15]. Specifically, we make use of ViSP's markerless generic model-based tracker for use with our RGB camera. To generate a measurement of the client vehicle's position and orientation, the ViSP tracker takes in the CAD model of the client vehicle, an initial estimate of the client vehicle's position and a camera image with the client vehicle present in the image frame; see Fig. 4.1.

4.1.2 Transforming Tracker Measurements to Inertial Frame

The ViSP tracker produces measurements of the client vehicle's pose with respect to the image frame of the camera mounted on the MRV base (see frame B in Fig. 4.2). Our Kalman Filter and Operational Space Controller operate in the inertial frame (see frame W in Fig. 4.2). To get the measured pose of the client vehicle in the inertial frame, we must apply the following transforms:

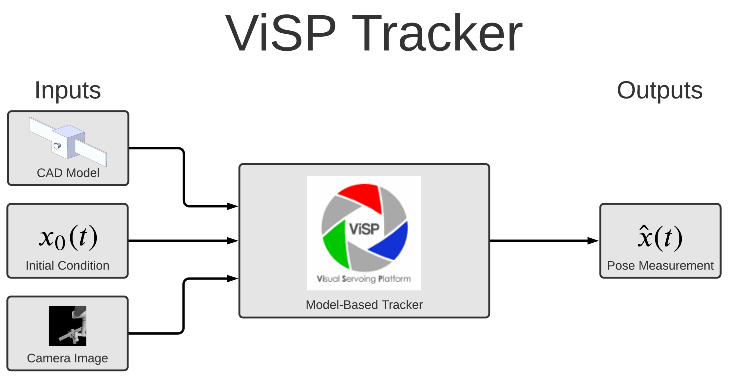


Figure 4.1: ViSP tracker block diagram

$$T_{WC} = T_{WA}T_{AB}T_{BC} \tag{4.1}$$

Where T_{WA} is given to us by a GPS position fix from the MRV, T_{AB} is known through the geometry of the spacecraft, and T_{BC} is published by the ViSP tracker

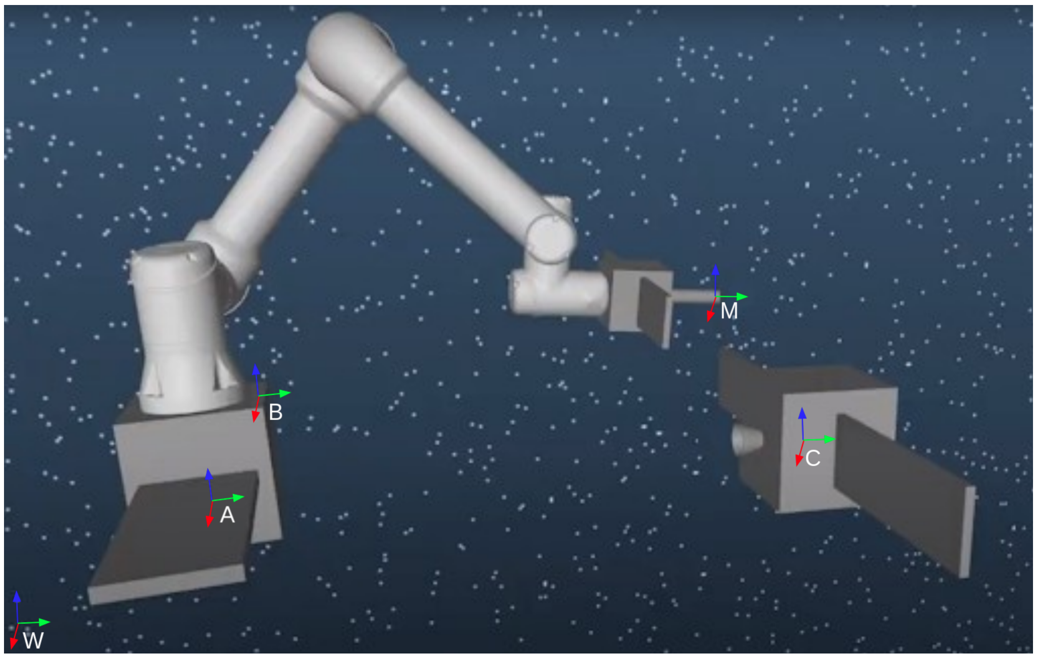


Figure 4.2: Overview of key coordinate frames in system. The world frame (W), the MRV base frame (A), the camera frame (B), the MEP frame (M), and the client vehicle frame (C).

4.2 Filtering

4.2.1 Hybrid Extended Kalman Filter (H-EKF) Overview

The ViSP tracker provides an *estimated* measurement of the pose of the client vehicle. The accuracy of this estimate is influenced by camera noise, variable lighting conditions, possible occlusions, and the presence of stars in the image background. To optimally estimate the state of the client vehicle, we implement a hybrid extended Kalman filter (H-EKF). We use a hybrid extended Kalman filter rather than a traditional extended Kalman filter because during operations, the underlying dynamics of our system change from when there is no contact, through contact with the client satellite, and after contacting the client satellite. To accommodate the switching dynamics, our H-EKF updates it's dynamics once contact between the MEP and client vehicle is detected. Below, we describe the updated equations of our H-EKF:

$$x_k = f(x_{k-1}, u_{k-1}) + \omega_{k-1} \tag{4.2}$$

Where x_k is the state of the client vehicle at time-step $t = k$, $f(x_{k-1}, u_{k-1})$ is the discrete process model state transition function, and ω_{k-1} is the process model zero mean Gaussian white noise at time-step $t = k$.

Our measurement model is described as follows:

$$z_k = h(x_{k-1}) + \nu_{k-1} \tag{4.3}$$

Where z_k is the measured state of the client vehicle (given to us by the ViSP tracker) at time-step $t = k$, $h(x_{k-1})$ is the measurement function which returns the current measurement, z_k, given the previous state, x_{k-1}. ν_{k-1} is the zero mean Gaussian white noise associated with the measurement function. The covariance matrices associated with the process model noise, ω_{k-1}, and measurement model noise, ν_{k-1}, are denoted Q, and R, respectively.

There are two steps to our H-EKF; prediction and update. Our prediction step is described by the following equations:

The predicted state estimate:

$$\hat{x}_k^- = f(\hat{x}_{k-1}^+, u_k - 1) \tag{4.4}$$

The predicted error covariance:

$$P_k^- = F_{k-1} P_{k-1}^+ F_{k-1}^T + Q \tag{4.5}$$

Our update step is described by the following equations:
The measurement residual:

$$\tilde{y}_k = z_k - h(\hat{x}_k^-) \tag{4.6}$$

The Kalman gain:

$$K_k = P_k^- H_k^T (R + H_k P_k^- H_k^T)^{-1} \tag{4.7}$$

The updated state estimate:

$$\hat{x}_k^+ = \hat{x}_k^- + K_k \tilde{y}_k \tag{4.8}$$

And the updated error covariance:

$$P_k^+ = (I - K_k H_k) P_k^- \tag{4.9}$$

Here, the ˆ operator denotes an estimate of a variable. The - operator represents predicted, or prior estimates. The + operator represents updated, or posterior estimates. F_{k-1} is given to us by taking the jacobian of the state transition function w.r.t. the state evaluated at $\hat{x}_{k-1}^+$ and u_{k-1}:

$$F_{k-1} = \frac{\partial f}{\partial x}|_{\hat{x}_{k-1}^+, u_{k-1}} \tag{4.10}$$

H_k is given to us by taking the jacobian of the measurement function w.r.t. the state and evaluating it at $\hat{x}_k^-$:

$$H_k = \frac{\partial h}{\partial x}|_{\hat{x}_k^-} \tag{4.11}$$

Now, we distinguish our estimator as a Hybrid Extended Kalman Filter because the underlying motion model, or process model, of our system switches when the MRV faces an external contact force. We can infer what force/torque is imparted onto the client by measuring the location of the contact force, as well as the magnitude/direction of the force felt at the MRV wrist. See Fig. 4.3

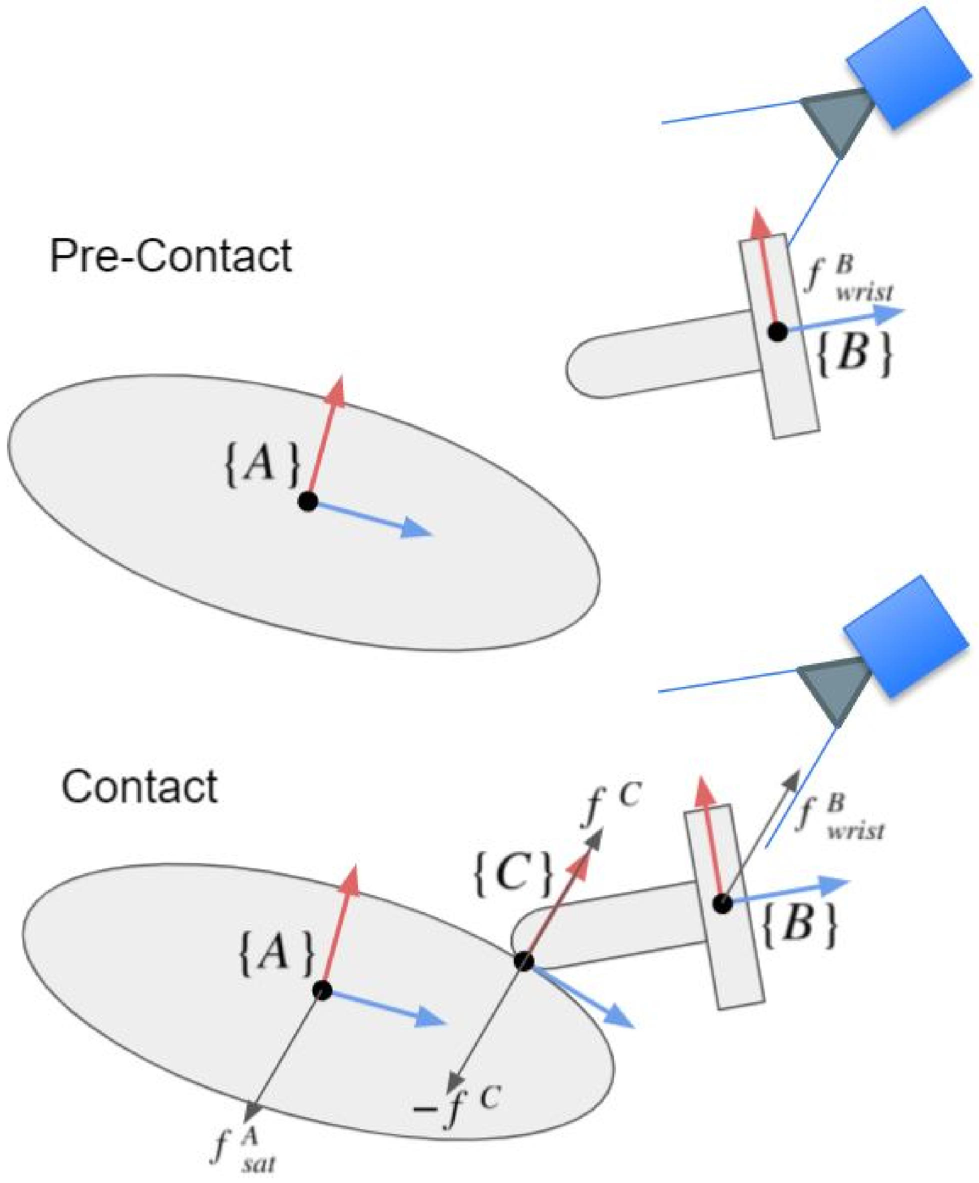


Figure 4.3: By leveraging the geometry of our MEP hook, we can determine the location of the contact of the client vehicle, as well as the resultant wrench applied to the body frame of the client vehicle.

Chapter 5

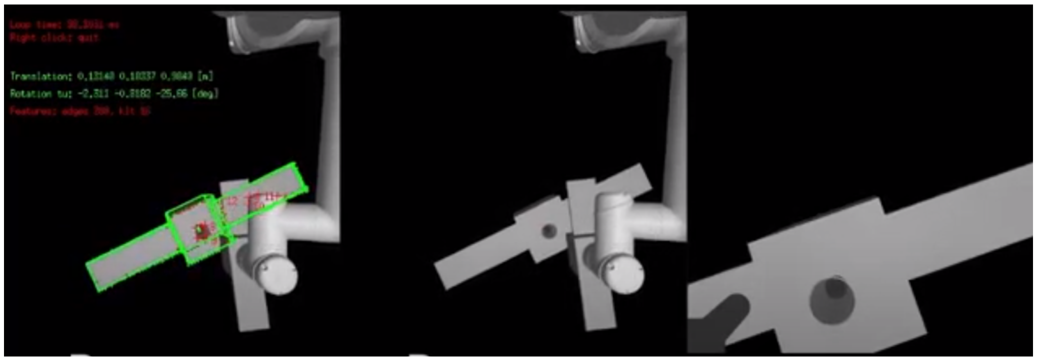
Integrated System

Here, we present results related to the performance of our integrated system. The integrated system features all of the sub-components presented above working together simultaneously (tracker, filter, and controller). To demonstrate the performance of our integrated system, we perform hundreds of experiments at multiple different initial conditions. Fig. 5.1 shows several camera perspectives during an integrated docking test. Table 5.1 shows a summary of the integrated system's performance at several initial conditions. Tables 5.2 and 5.3 show the initial conditions that were used for our test. As can be seen from Table 5.1, the integrated tracking, filtering, and control system performs quite well across the five batches of 100 trials that were run.

Integrated Test Summary					
Batch	Avg. Success (%)	Avg. Error, X (mm)	Avg. Error, Y (mm)	Avg. Error, Z (mm)	Avg. Error, Rotation (rad)
1	97.6± 1.14	7 ± 5	2 ± 8	6 ± 6	.01 ± .02

Table 5.1: Performance summary of integrated system across three different initial conditions. Descriptions of initial conditions can be found in Table 5.2. Average success and error calculations were computed across 5 sets of 100 trials for each initial condition scenario.

Each trial of our integrated systems performance test consists of two steps: First, the approach step, where the MEP is brought directly in front of the client vehicle's



```
Loop time: [illegible] ms
Right click: quit

Translation: 0.13148 0.10337 0.9843 [m]
Rotation tu: -2.311 -0.3182 -25.66 [deg]
Features: edges 288, klt [illegible]
```

Figure 5.1: Three separate camera views from the MRV. Left: the ViSP tracker's belief state overlaid onto the camera view from the MRV deck plate. Middle: The MRV base camera view without a ViSP tracker overlay. Right: The view from the simulated camera mounted at the front of the MEP.

Integrated Test Initial Conditions - Client Vehicle Pose						
Batch	CV X (m)	CV Y (m)	CV Z (m)	CV Roll (rad)	CV Pitch (rad)	CV Yaw (rad)
1	0.8	0.1	0.2	0.65	0.3	0

Table 5.2: Relevant position and Euler angles of the Client Vehicle for each integrated test.

rocket nozzle. Once the MEP is within 1.5cm of the final approach waypoint (see blue dot in Fig. 5.2, then the system enters the dock step, where the target waypoint is moved into the client satellite's rocket nozzle. Docking is complete once the MEP reaches within five millimeters of the final docking waypoint inside of the client vehicle's nozzle. Fig. 5.3 shows the overlaid plots of the raw measurements coming from ViSP, the filtered measurement returned by the H-EKF, and the ground truth measurement provided by our simulation environment.

Integrated Test Initial Conditions - Arm Joint Angles						
Batch	θ_1 (rad)	θ_2 (rad)	θ_3 (rad)	θ_4 (rad)	θ_5 (rad)	θ_6 (rad)
1	-0.304	-0.543	1.941	-1.402	1.264	-1.565

Table 5.3: Manipulator arm joint angles for each integrated test.

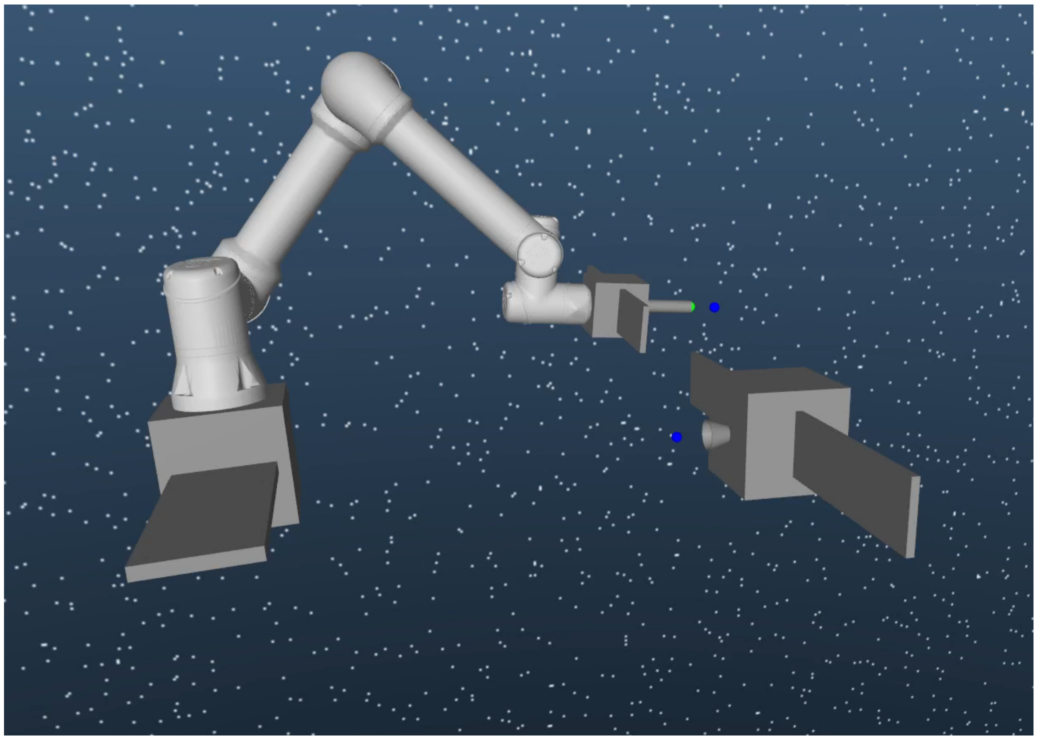


Figure 5.2: Third person view of integrated system test. The blue dot outside the client vehicle represents the target waypoint for the approach stage.

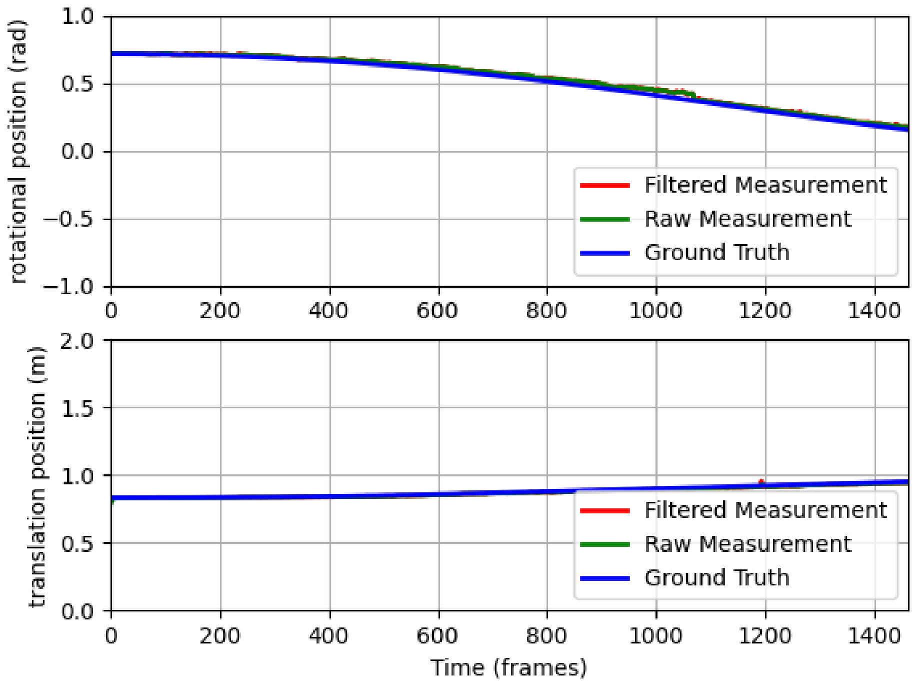


Figure 5.3: Time history of normalized translational and rotational positions for an individual integrated system trial. Note that the raw and filtered measurements are nearly on top of the ground truth signal.

Chapter 6

Future Work

This work could be improved and extended in several ways:

6.0.1 Two Arm Manipulation

One extension that could be made would be to enable our operational space controller to control two manipulator arms simultaneously. This would have direct applications for the NGC MRV, which is equipped with two manipulator arms. This could enable more complicated repair and maintenance activities, such as tasks which involve the use of more than one tool, or a construction task requiring the mating of two separate bodies.

6.0.2 Moving/Tumbling Client Satellite

Another extension to our work would be to test and validate the performance of our docking pipeline on a moving client satellite. Our current work assumes an unactuated, static client vehicle. A moving target would be more challenging because our controller would have to match both the position and velocity of the client vehicle. On top of matching the position/velocity of the rocket nozzle of the moving client vehicle, we would also need to begin a maneuver to cancel out the existing linear and angular momentum of the client vehicle. This would involve additional motion planning effort. Additionally, we would need to perform tests to verify the maximum translational and rotational velocities that our control system can handle while still guaranteeing a successful capture.

6.0.3 Controller Optimization

We could improve the performance of our controller by performing an optimization routine over the OSC controller gains. This optimization could be done via a brute force search, a Monte-Carlo search, or by leveraging existing Bayesian optimization to tune our controller gains.

6.0.4 Robust Client Satellite Tracking

Our current solution for client vehicle tracking is to use an off the shelf solution, the ViSP tracker. This tracker performs well in situations without occlusion, but it struggles somewhat in situations with occlusions. There are several possible approaches to improve the quality of tracking under occlusion, such as incorporating the image stream from a depth camera, and using existing state-of-the-art search-based pose estimation routines [3], or alternatively leveraging existing state of the art learning based approaches which have been shown to gracefully handle occlusions [23].

6.0.5 Robust Motion Planning

Our work could be extended by adding a more thorough motion planning step to the control pipeline. Although our control system uses force feedback to help guide the MEP into the client vehicle's rocket nozzle, there are situations where a motion planner would be necessary in order to guarantee the successful docking of the MEP with the client vehicle. Example extensions here would be to incorporate the use of a SLAM system to map out obstacles in the environment, and an A^* or RRT planner to plan a feasible trajectory through the environment.

6.0.6 Robustness Analysis

Additional focus can be emphasized on studying the influence of varying environmental conditions on the operation of our control system. Changes to the environment, such as varying system and component operation frequencies, changes in lighting, changes in the image background (i.e. stars, moon, and sun), and changes in image noise all have the potential to influence the performance of our control system. Studies investigating the effects of all of these environmental changes could be run in our existing simulation environment.

6.0.7 Scaling Adjustments

Future work could focus on refining the scale and geometry of the simulation environment to more faithfully represent the true space system. Currently, some of the dimensions of the bodies in our environment are not perfectly matched with the dimensions of the corresponding bodies launched into space. Additionally, our current simulated MRV system features a 6-DOF arm whereas the true space system features a 7-DOF manipulator arm. Additional effort could be expended to reduce the number of geometric differences between our simulated system and the true space system deployed by NGC.

6.0.8 Hardware in the Loop Testbed

Lastly, our work here does not make any efforts to perform real life robot experiments. Several groups have deployed their control systems on gravity assisted platforms, such as air tables, and other frictionless environments, in order to simulate space-like zero gravity conditions in the plane. Future work could involve developing a hardware in the loop testbed system to test the efficacy of our control, tracking, and estimation algorithms on real robots.

An alternative route would be to develop a test rig featuring emulated spacecraft mounted on 6-DOF travels or robot arms. With such a system, we could "simulate" the space dynamics by actuating the relevant bodies using the 6-DOF travels based on information from control inputs and contacts.

Chapter 7

Conclusions

In conclusion, we presented our work on the development and analysis of a simulation environment and control system for a floating base manipulator to dock with a client satellite and deliver a life extension payload. We also presented a brief history of space robots, and an overview of existing control and tracking methods. The integrated system analysis we performed shows that our estimation and control pipeline enables successful docking missions from a variety of initial conditions with high confidence. Our analysis of the vision tracking subsystem tells us that more work is needed in reducing the uncertainty caused by occlusions in the client satellite pose estimate.

A key result we displayed was our extension of prior work done in operational space control. We demonstrated through exhaustive numerical experiments that contact force information can be used to enable an operational space control routine to operate in a partially observable environment. An additional contribution of this work was to create a simulation and control pipeline for space manipulators to enable the rapid development of new research on control and motion planning for space manipulators.

Bibliography

[1] *Planar Motion Control, Coordination and Dynamic Entrainment in Chaplygin Beanies*, Dynamic Systems and Control Conference, 09 2018. doi: 10.1115/DSCC2018-9037. URL https://doi.org/10.1115/DSCC2018-9037. V003T40A007. 2.1

[2] Spacelogistics, 2019. URL https://www.northropgrumman.com/space/space-logistics-services/. 2.1

[3] Aditya Agarwal, Yupeng Han, and Maxim Likhachev. Perch 2.0: Fast and accurate gpu-based perception via search for object pose estimation. In *Proceedings of (IROS) IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 10633 – 10640, October 2020. 2.4, 6.0.4

[4] Morten F. Amundsen, Jørgen Sverdrup-Thygeson, Eleni Kelasidi, and Kristin Y. Pettersen. Inverse kinematic control of a free-floating underwater manipulator using the generalized jacobian matrix. In *2018 European Control Conference (ECC)*, pages 276–281, 2018. doi: 10.23919/ECC.2018.8550525. 1

[5] Martin Behrendt. *A basic working principle of Model Predictive Control*. 2009. URL https://commons.wikimedia.org/wiki/File:MPC_scheme_basic.svg. (document), 3.5

[6] Monimoy Bujarbaruah, Xiaojing Zhang, Eric Tseng, and Francesco Borrelli. Adaptive mpc for autonomous lane keeping. 02 2018. 3.1

[7] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013. 3.1.2

[8] D'Errico. MATLAB adaptive robust numerical differentiation. https://www.mathworks.com/matlabcentral/fileexchange/13490-adaptive-robust-numerical-differentiation, 2020. Accessed: 2020-12-16. 3.1.2

[9] Jaskaran Grover, Jake Zimmer, Tony Dear, Matthew Travers, Howie Choset, and Scott David Kelly. Geometric motion planning for a three-link swimmer in a three-dimensional low reynolds-number regime. In *2018 Annual American Control Conference (ACC)*, pages 6067–6074, 2018. doi: 10.23919/ACC.2018.8431828.

2.1

[10] Jaskaran Grover, Daniel Vedova, Nalini Jain, Matthew Travers, and Howie Choset. Motion planning, design optimization and fabrication of ferromagnetic swimmers. In *Proceedings of Robotics: Science and Systems (RSS '19)*, pages 79 – 87, June 2019. 2.1

[11] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987. doi: 10.1109/JRA.1987.1087068. 2.3

[12] Nathan J. Kong, J. Joe Payne, George Council, and Aaron M. Johnson. The salted kalman filter: Kalman filtering on hybrid dynamical systems. *Automatica*, 131:109752, 2021. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2021.109752. URL `https://www.sciencedirect.com/science/article/pii/S0005109821002727`. 2

[13] Robert E. Lindberg, Richard W. Longman, and Michael F. Zedd. *Kinematic and Dynamic Properties of an Elbow Manipulator Mounted on a Satellite*, pages 1–25. Springer US, Boston, MA, 1993. ISBN 978-1-4615-3588-1. doi: 10.1007/978-1-4615-3588-1_1. URL `https://doi.org/10.1007/978-1-4615-3588-1_1`. 2.2

[14] Richard W. Longman. *The Kinetics and Workspace of a Satellite-Mounted Robot*, pages 27–44. Springer US, Boston, MA, 1993. ISBN 978-1-4615-3588-1. doi: 10.1007/978-1-4615-3588-1_2. URL `https://doi.org/10.1007/978-1-4615-3588-1_2`. 2.2

[15] E. Marchand, F. Spindler, and F. Chaumette. Visp for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics Automation Magazine*, 12(4):40–52, 2005. doi: 10.1109/MRA.2005.1577023. 2, 2.4, 4.1.1

[16] Matthew T. Mason. Compliance and force control for computer controlled manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(6): 418–432, 1981. doi: 10.1109/TSMC.1981.4308708. 1.2

[17] Mathworks. MATLAB constrained nonlinear optimization. mathworks.com/help/optim/index.html, 2020. Accessed: 2020-12-16. 3.1.2, 3.1.2

[18] Mitsushige Oda, Kouichi Kibe, and Fumio Yamagata. Ets-vii, space robot in-orbit experiment satellite. In *Proceedings of IEEE international conference on robotics and automation*, volume 1, pages 739–744. IEEE, 1996. 2.1

[19] Andrew Packard, Roberto Horowitz, Kameshwar Poolla, and Francesco Borrelli. ME 132, Dynamic Systems and Feedback. page 493, 2018. 3.1.2

[20] Ludovic Righetti, Jonas Buchli, Michael Mistry, and Stefan Schaal. Inverse

dynamics control of floating-base robots with external constraints: A unified view. In *2011 IEEE International Conference on Robotics and Automation*, pages 1085–1090, 2011. doi: 10.1109/ICRA.2011.5980156. 2.3

[21] Tomasz Rybus, Karol Seweryn, and J. Sasiadek. Control system for free-floating space manipulator based on nonlinear model predictive control (nmpc). *Journal of Intelligent Robotic Systems*, 85, 03 2017. doi: 10.1007/s10846-016-0396-2. 3.1, 3.1, 3.1.3

[22] L. Sentis and O. Khatib. Control of free-floating humanoid robots through task prioritization. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1718–1723, 2005. doi: 10.1109/ROBOT.2005.1570361. (document), 1, 2.3, 2.3, 3.2.2

[23] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. In *Conference on Robot Learning (CoRL)*, 2018. URL `https://arxiv.org/abs/1809.10790`. 2.4, 6.0.4

[24] Z. Vafa and S. Dubowsky. *On the Dynamics of Space Manipulators Using the Virtual Manipulator, with Applications to Path Planning*, pages 45–76. Springer US, Boston, MA, 1993. ISBN 978-1-4615-3588-1. doi: 10.1007/978-1-4615-3588-1_3. URL `https://doi.org/10.1007/978-1-4615-3588-1_3`. 2.2

[25] Joseph Virgili-Llop et al. SPART: an open-source modeling and control toolkit for mobile-base robotic multibody systems with kinematic tree topologies. `https://github.com/NPS-SRL/SPART`, 2018. 3.1, 3.1

[26] G. Visentin and D.L. Brown. Robotics for geostationary satellite servicing. *Robotics and Autonomous Systems*, 23(1):45–51, 1998. ISSN 0921-8890. doi: https://doi.org/10.1016/S0921-8890(97)00057-2. URL `https://www.sciencedirect.com/science/article/pii/S0921889097000572`. Space Robotics in Europe. 1.1

[27] CG Wagner-Bartak, JA Middleton, and JA Hunter. Shuttle remote manipulator system hardware test facility. In *11th Space Simulation Conference*, volume 2150, page 79, 1980. 2.1

[28] Markus Wilde, Stephen Kwok Choon, Alessio Grompone, and Marcello Romano. Equations of motion of free-floating spacecraft-manipulator systems: An engineer's tutorial. *Frontiers in Robotics and AI*, 5:41, 2018. ISSN 2296-9144. doi: 10.3389/frobt.2018.00041. URL `https://www.frontiersin.org/article/10.3389/frobt.2018.00041`. (document), 2.2, 2.2

[29] M. Wüthrich, P. Pastor, M. Kalakrishnan, J. Bohg, and S. Schaal. Probabilistic object tracking using a range camera. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3195–3202. IEEE, November 2013. 2.4

[30] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes, 2018. 2.4

[31] Kazuya Yoshida and Yoji Umetani. *Control of Space Manipulators with Generalized Jacobian Matrix*, pages 165–204. Springer US, Boston, MA, 1993. ISBN 978-1-4615-3588-1. doi: 10.1007/978-1-4615-3588-1_7. URL https://doi.org/10.1007/978-1-4615-3588-1_7. 2.2, 2.3