ENPM662: Introduction to Robot Modeling

Final Project

A Project Report on

Modeling and Simulation of Satellite Servicing Manipulators

Adarsh Malapaka
UID: 118119625

Sai Sandeep Adapa
UID: 118411460

Date of Submission: 11^{th} *December, 2021*

Google Drive Project Files:
`https://drive.google.com/drive/folders/1vWNAVEqq27iOx6S-vtfcc-tpMvRrbiVr?usp=sharing`

A. James Clark School of Engineering (Robotics)
University of Maryland, College Park, MD - 20742

Table of Contents

Appendices

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Satellite servicing on orbit is complicated by several factors primarily because nearly all satellites don't have the ability to service themselves. The issue of serviceability is among the challenges this raises. It involves cutting and partially removing the protective thermal blanketing prior to servicing a satellite operation. Earth-based operators can handle this task through telerobotics, but they will need to overcome some challenges posed by satellite technology. Long telemetry delays between satellites are one such challenge. And the operator and the limited, or even obstructed, views from the available cameras. Despite the limitations, utilizing satellite servicing robots can bring down the cost of a space mission by enhancing the orbital lifetime of the designated satellites. [1]

The applications are not only limited to Satellite Servicing but can also be extended to implement space junk cleaning missions. Our orbit is filled with debris, including chunks of dead satellites, discarded rockets, and paint flecks that have fallen off them. A part of the presented Satellite servicing model can be used to latch onto debris before diving back down to Earth, where both machine and junk will burn up in the atmosphere.

The world-famous Hubble Telescope launched on April 24, 1990, had to undergo 5 Service missions in its lifetime. The service cost for each mission was humongous and required astronauts to risk spacewalks to repair the telescope. For instance, Servicing Mission 3 was divided into two flights—Servicing Mission 3A (SM3A), and Servicing Mission 3B (SM3B). The cost to carry out the 3A mission was $136 million. This included $19 million of HST costs for the additional servicing mission, $7 million of HST costs to switch from Columbia to Discovery, and $110 million for the Shuttle to carry out the mission and replace Space Shuttle flight hardware previously assigned to another mission. [2] [3] NASA had also spent approximately $69 million on Servicing Mission 3A, in addition to the $136 million, reflecting the costs

of building and testing the planned replacement hardware, ground operations, and other related activities. The proposed project can reduce the cost of satellite service dramatically and also avoid human intervention thereby saving lives.

In 1996, a French satellite was hit and damaged by debris from a French rocket that had exploded a decade earlier. On Feb.10, 2009, a defunct Russian spacecraft collided with and destroyed a functioning U.S. Iridium commercial spacecraft. The collision added more than 2,300 pieces of large, trackable debris and much smaller debris to the inventory of space junk. China's 2007 anti-satellite test, which used a missile to destroy an old weather satellite, added more than 3,500 pieces of large, trackable debris and much smaller debris to the debris problem. The implemented telerobot in this project can be used as a pragmatic solution to clear space debris as mentioned in the Introduction section.

1.2 Application

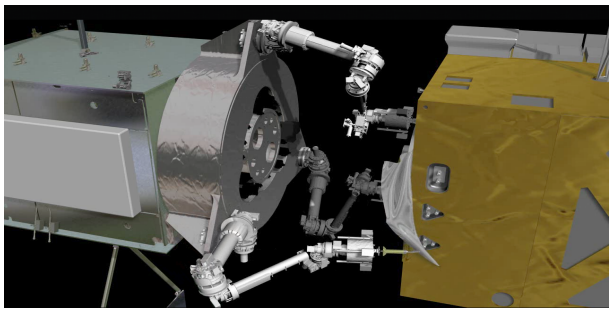

Figure 1.1: Servicing satellite with 2 robot arms (left) and Satellite to be serviced (right)

Source: https://nexis.gsfc.nasa.gov/

The primary application of the servicing robot is to perform servicing operations on the repairable satellites such as cutting the Multi-Layer Insulation (MLI) 1.2 thereby enabling replacing/repairing of the satellite subsystems allowing satellites to work longer in space. [4]

Every year, functional satellites providing weather data, communications and other essential services are retired because they reach the end of their fuel supply. As old or ailing satellites are moved to the "graveyard orbit," re-

Figure 1.2: Multi-Layer Insulation (MLI) on Satellites
Source: https://aerospacefab.com/products/multilayer-insulation/

placement models are financed, designed, built and launched—a process that can be costly and time consuming for satellite owners and the consumers. To begin with, the fuel valves on today's satellites were never intended to be accessed after launch. Satellite makers use wired-shut caps to tightly lock down the fuel valve during launch and operations to prevent hazardous fuel from leaking.The bulk of satellites in geosynchronous Earth orbit that could benefit from servicing are in this orbit. GEO is now inaccessible to humans, which means that only robots can complete the task. The presented ERA can be modified to match the technology of the Robotic Refueling Mission. [5]

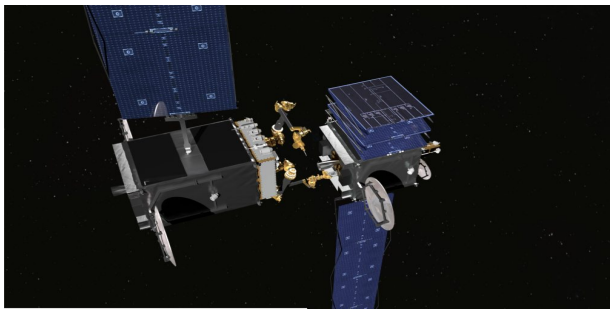

Figure 1.3: Servicing Satellite

Chapter 2

Robot Modeling

2.1 Design Reference

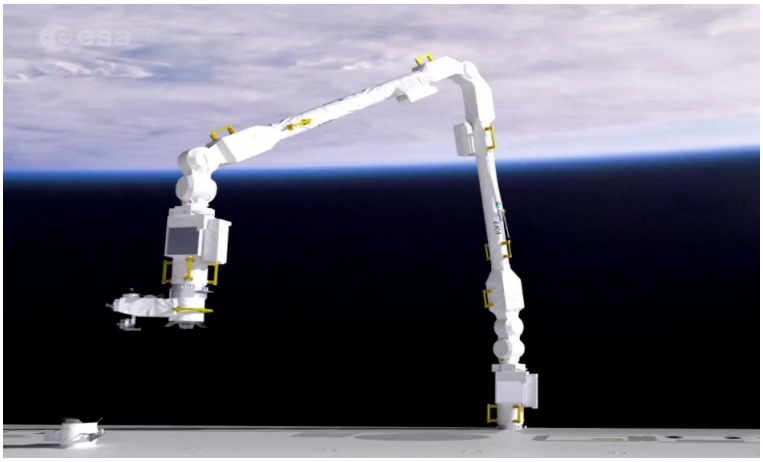

Figure 2.1: European Robotic Arm (ERA)
Source: https://www.esa.int

The European Robotic Arm (ERA) [Fig. 2.1] is attached to the Russian Orbital Segment of the International Space Station (ISS) in outer space and was developed for the European Space Agency (ESA) to be used by the astronauts in the maintenance of the Space Station in addition to acting as a tool to transfer small payloads from inside to outside the ISS. The ERA acts as a tool for: installation, deployment and replacement of elements of the Russian Segment of the Space Station, inspection of the Russian Segment, support/transfer of EVA cosmonauts, transfer of Orbital Replacement Units and other assembly tasks. The arm consists of 2 End Effectors, 2 Wrists, 2 Limbs and 1 Elbow joint together with electronics and cameras. Both ends act as either a "hand" for the robot or the base from which it can operate.

The symmetrical design of the arm with a complete 3 degree-of freedom

wrist and general-purpose end effector on both sides, allows the ERA to relocate on the station by grappling a new base point and releasing the old one, and move to different working locations. It has been designed from the beginning to be serviceable and to allow exchange of big parts via EVA in case of failure or an emergency situation. Such parts are called ERU (EVA Replaceable Units). Its seven rotational joints – roll, yaw and pitch joints – provide a larger range of motion than a human arm. The robot can walk end-over-end grappling standard fixtures of the Russian segment in an inchworm-like movement. The limbs are designed for efficiency. A carbon fiber-wound tube is used to minimise mass and has titanium fittings bonded onto it. Strength and flexibility are key for this spacewalker. [10] [11]

Table 2.1: Specifications of the ERA robot [11]

Property	Value
Arm Length (m)	11.3
Maximum Reach (m)	9.7
Materials	Carbon-fiber & Aluminium
Degrees of Freedom	7 (each arm)
Sensors	Cameras, Force/Torque Sensor (end-eff)
Mass (kg)	630
Max. Payload (kg)	8000
Max. Payload dimensions (m)	3 x 3 x 8.1
Motors (each arm)	7 BLDC motors
Power Consumption, peak (W)	800
End-Effector speed, max (m/s)	0.1
Accuracy (m)	0.005

2.2 Robot Description

2.2.1 Type & Degrees of Freedom

In this project, the previously discussed ERA robot arm which is a robot arm manipulator, is modeled and simulated along with the satellite system to constitute the satellite servicing setup. The robotic system consists of a 'servicing' satellite on which is mounted 1 robot arm manipulator with a total of 6 degrees of freedom. The base of the manipulator along with the wrist, which are fixed on to the satellite, consist of 3 revolute joints to provide a wide range of motion. This 3 DOF wrist is then succeeded by an

elbow joint containing 1 revolute joint in the kinematic chain. Finally, at the end effector, a 2 degree of freedom wrist is present for the end-effector to be able to reach any configuration point on the satellite to be 'serviced', in outer space. This is in contrast to the actual ERA arm that consists of 7 degrees of freedom with a 3 degree of freedom wrist. The end of the robot arm consists of a rotary cutting blade apparatus to cut open the multi-layer insulation (MLI) on the satellite to be serviced. The robot manipulator, both the original ERA arm and the custom modeled arm, contain visually symmetrical parts such that it becomes easier to replace components of the arm, in the event of the arm itself requiring maintenance.

2.2.2 Dimensions

The dimensions of our custom ERA robot were scaled down from the high values from the actual system. This was done to be able to fit the system into our Gazebo simulation world (discussed later) and to create a miniaturized version of the actual system. Also, the scaling down was done based on the satellite CAD model it was mounted to. [Fig. 2.2] shows the actual lengths of the ERA system for each link. This can be compared with the individual link lengths taken in our custom model as shown in Table 2.2. This table also contains the mass of each individual link of our robot arm. It is seen that the total length of the actual system is approximately 11.3 meters whereas our model is nearly 5 meters long. The total mass of the actual system is nearly 630 kg whereas our model weighs around 96 kg.

Images of the dimensions obtained from the CAD models of each part file of our custom ERA robot can be referred to for more details.[1]

[1]Link to images of dimensions of custom ERA model - `https://drive.google.com/drive/folders/1wx8Xf5B6GFk9YXmR8TjD8fawsbBwKheT?usp=sharing`

Table 2.2: Specifications of the custom-designed ERA arm

Link	Length (mm)	Mass (kg)
Base	577	4.0606
Lower Wrist	224	1.572
Lower Limb	1572	10.7876
Upper Limb	1572	9.9015
Upper Wrist	224	1.5257
End-Effector	577	4.0606
Cutter Tool	260	0.0192
Total	**5006**	**31.9272**

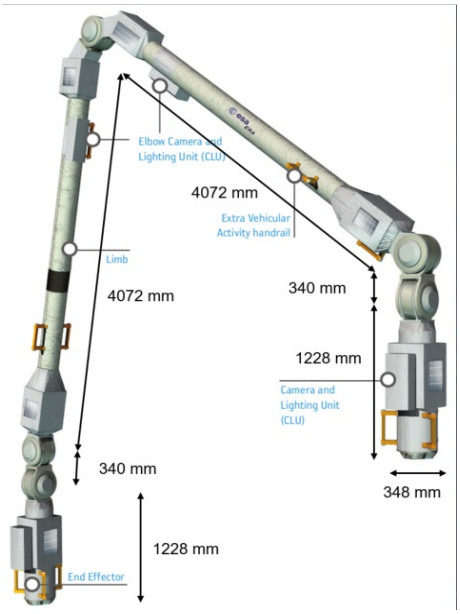

Figure 2.2: Dimensions of the actual ERA Robot Arm [12]

2.3 CAD Models

Based on the design of the ERA robot arm discussed before, the CAD models of the robot arm were obtained using SolidWorks [Fig. 2.4 to 2.8]. Due to the symmetry of the robot arm, the lower wrist and base CAD models were reused for the upper wrist and end-effector CAD models. Although the lower and upper limb links are symmetrical, two different CAD models were

made to accommodate the mutual joint surface on the lower limb model. The cutter tool model consists of a black cuboid to represent a motor and a gray blade as the rotary cutter of the cutting apparatus. Pegs were provided for easy mating to the end-effector model.

[Fig. 2.3] shows the assembled ERA arm CAD model from the individual parts and [Fig. 2.9] shows the entire assembled servicing satellite setup. The satellite CAD model was taken from the internet from a design of a Turkish satellite by *Haktan Yağmur* [13].[2]

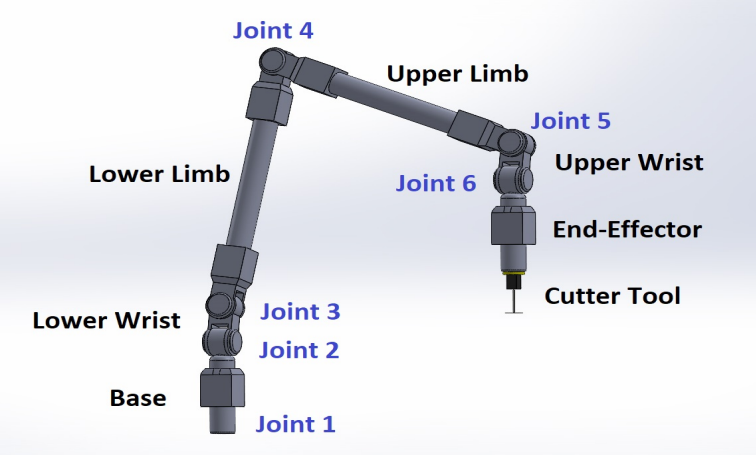

Figure 2.3: ERA Robot Arm CAD Assembly

[2]Satellite Servicing CAD Parts & Assemblies:
`https://drive.google.com/drive/folders/1ObKWCvImXbVH9s-wlTQG4Dd58iBcpSdJ?usp=sharing`

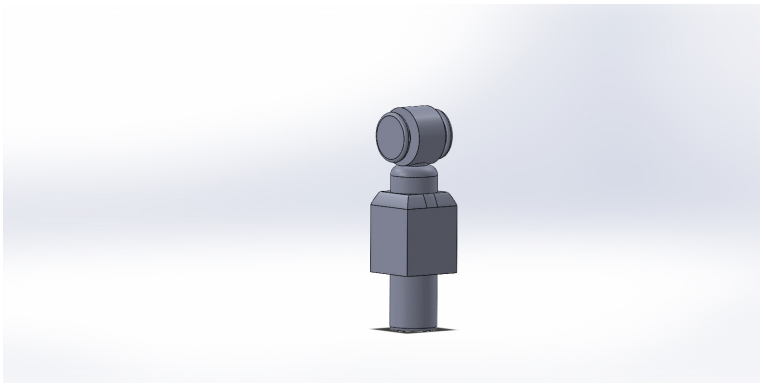

Figure 2.4: Base/End-Effector Link CAD Part

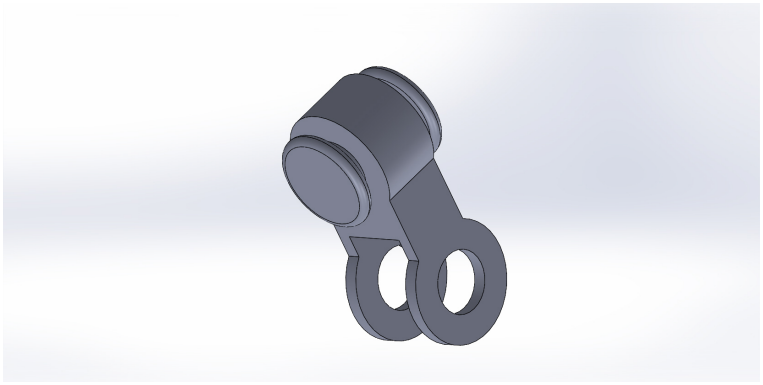

Figure 2.5: Wrist (Upper/Lower) CAD Part

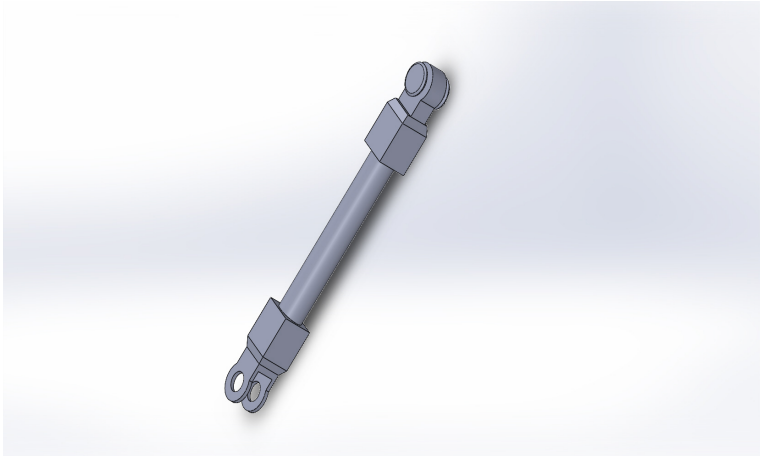

Figure 2.6: Lower Limb CAD Part

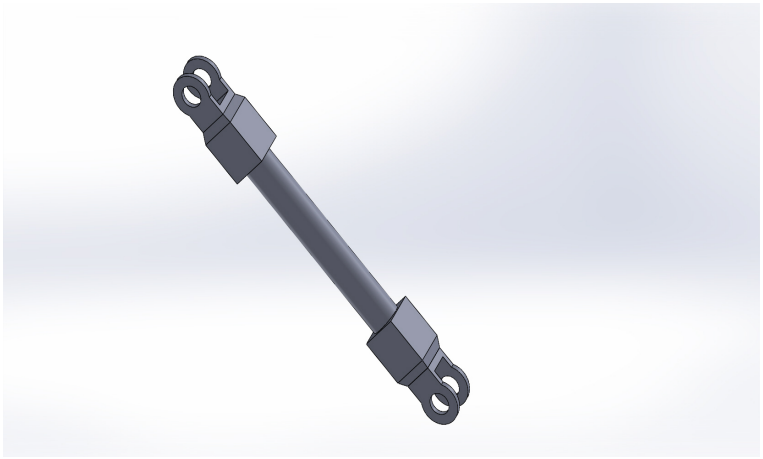

Figure 2.7: Upper Limb CAD Part

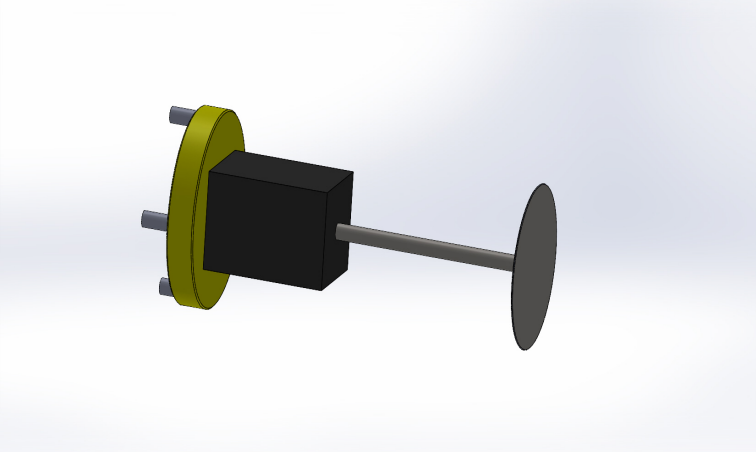

Figure 2.8: Cutter Tool CAD Part

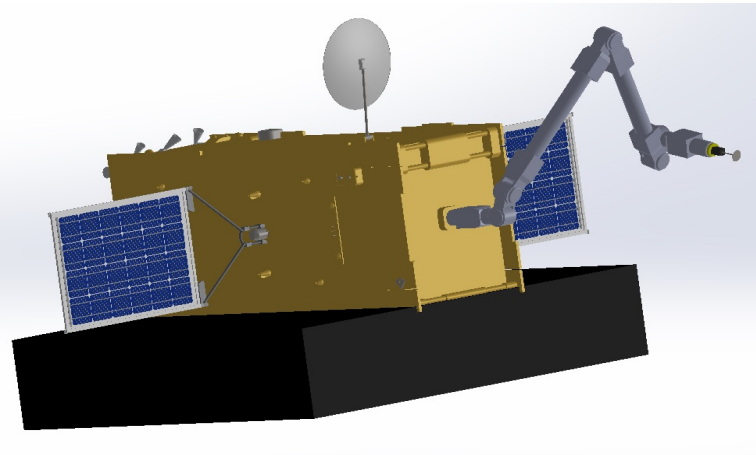

Figure 2.9: Servicing Satellite Assembly

2.4 Assumptions

Since our application is on the usage of robot manipulators in outer space to perform tasks that assist astronauts and service old satellites, certain assumptions have been made to strike a fine balance between physical reality and ideal behaviour in order to simplify the problem statement for the simulation to run effectively.

- All links and bodies in the system are assumed to be rigid bodies with time-independent kinematic and dynamic parameters such as masses and lengths.
- The friction and damping effects at the joints of the manipulator have been neglected or assumed to be very low.
- The joint axes of the actuators on the robot arm have been assumed to be perfectly aligned with the general coordinate frame Z axes (assigned in the next chapter) with no misalignment due to manufacturing tolerances.
- Although the original ERA robot system has 7 degrees of freedom, one degree of freedom in the wrist has been eliminated to bring the toal degrees of freedom to 6 in order to achieve numerical stability of the solution of the inverse kinematics and to obtain a square Jacobian matrix which is invertible.
- The servicing satellite and satellite to be serviced have been fixed to the simulation world's ground with zero relative velocity between the two. This has been done to emulate the arresting of relative velocity between the two satellites after the *servicer* docks onto the recipient satellite using one of its robot arms.
- The effect of gravity and friction (wherever necessary) shall be neglected or assumed to be the value associated with that of the Low Earth Orbit (LEO). For simulation purposes, the gravity magnitude was taken as -2.5m/s^2 instead of complete 0 gravity to constrain unwarranted motions in the system due to small impulses while trajectory tracking.
- The round-trip communication telemetry delays between the Earth and Space has been assumed to be 5 seconds, whereas in general this value is typically anywhere between 5-7 seconds.

Chapter 3

Kinematics

3.1 Denavit-Hartenberg Convention

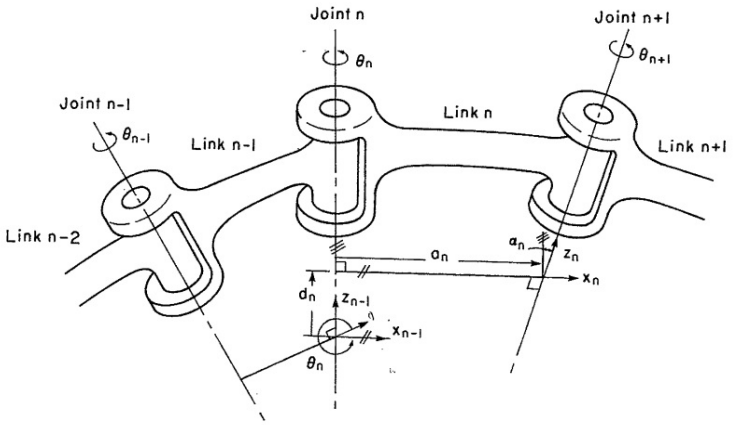

Figure 3.1: D-H Parameters for consecutive links [8]

The Denavit-Hartenberg (D-H) convention[3] [7] is widely used in robotics for fixing coordinate frames to rigid bodies such as links and kinematically describing them with the help of only four parameters. [Fig. 3.1] shows a picture of the definition of the joint angles and frame assignments on two consecutive links of a robot arm manipulator consisting of only revolute joints. The four D-H parameters are defined as:

- a_i: Distance from origin O_i to the intersection of the z_{i-1} and x_i axes along the x_i axis.
- d_i: Distance from origin O_{i-1} to the intersection of the z_{i-1} and x_i axes along the z_{i-1} axis.

[3]Note: In general, there are two types of D-H conventions: Standard & Modified. Here, only the standard D-H convention (Spong) has been used.

- α_i: Angle between z_{i-1} and z_i axes about the x_i axis.
- θ_i: Angle between x_{i-1} and x_i axes about the z_{i-1} axis.

Here, a_i is the Link Length, α_i is the Link Twist, d_i is the Link Offset and θ_i is the Joint Angle. Using the D-H parameters and assigned frames, the homogeneous transformation matrix (describing both rotation and translation between the two frames) between links $\{i–1\}$ and $\{i\}$ can be expressed as [8],

$$ {}^{i-1}_{i}T = \begin{bmatrix} cos\theta_i & -cos\alpha_i sin\theta_i & sin\alpha_i sin\theta_i & a_i cos\theta_i \\ sin\theta_i & cos\alpha_i cos\theta_i & -sin\alpha_i cos\theta_i & a_i sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1} $$

3.2 D-H Parameters & Table

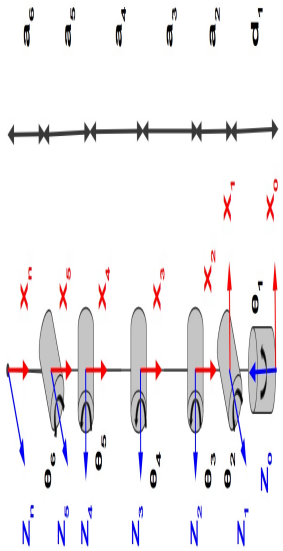

Figure 3.2: Kinematic diagram of the ERA robot (home position)

Using the D-H convention stated previously, the kinematic diagram of the ERA servicing robot arm when it is at it's home position (all joint angles zero) is obtained and depicted in 3.2. Each of the gray cylinders represents

the corresponding joint with the attached link co-ordinate frame. The X and Z axes of each frame are depicted and the Y axis completes the right-hand rule.

Table 3.1: D-H Table for ERA robot

Link	θ_i (rad)	d_i (m)	α_i (rad)	a_i (m)
1	θ_1^*	0.577	$\pi/2$	0
2	θ_2^* - $\pi/2$	0	$\pi/2$	-0.224
3	θ_3^*	0	0	-1.572
4	θ_4^*	0	0	-1.572
5	θ_5^*	0	-$\pi/2$	-0.224
n	θ_6^*	0	0	-0.837

In the D-H table of the robot shown in Table 3.1, Link 'n' represents the end-effector which also consists of the cutter tool (length: 0.26m) mounted on the robot arm's end-effector. Also, it is observed that that the a_i parameters are negative despite representing distances. This is because the distance is measured in a specific way as mentioned previously. There is an offset of 90 degrees in the joint angle, θ_2 of the robot as seen in the table.

3.3 Forward Kinematics

From the D-H table and [Eq. 3.1], the transformation matrices for each subsequent link pair of the robot arm are obtained as follows:

$$ {}^{0}_{1}T = \begin{bmatrix} cos\theta_1 & 0 & sin\theta_1 & 0 \\ sin\theta_1 & 0 & -cos\theta_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2} $$

$$ {}^{1}_{2}T = \begin{bmatrix} sin\theta_2 & 0 & -cos\theta_2 & a_2 sin\theta_2 \\ -cos\theta_2 & 0 & -sin\theta_2 & -a_2 cos\theta_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3} $$

$$ {}^{2}_{3}T = \begin{bmatrix} cos\theta_3 & -sin\theta_3 & 0 & a_3 cos\theta_3 \\ sin\theta_3 & cos\theta_3 & 0 & a_3 sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4} $$

$$ {}^{3}_{4}T = \begin{bmatrix} cos\theta_4 & -sin\theta_4 & 0 & a_4 cos\theta_4 \\ sin\theta_4 & cos\theta_4 & 0 & a_4 sin\theta_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5} $$

$$ {}^{4}_{5}T = \begin{bmatrix} cos\theta_5 & 0 & -sin\theta_5 & a_5 cos\theta_5 \\ sin\theta_5 & 0 & cos\theta_5 & a_5 sin\theta_5 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6} $$

$$ {}^{5}_{n}T = \begin{bmatrix} cos\theta_6 & -sin\theta_6 & 0 & a_6 cos\theta_6 \\ sin\theta_6 & cos\theta_6 & 0 & a_6 sin\theta_6 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7} $$

Using the compound multiplication property of transformation matrices, the transformation of the end of the cutter tool with respect to the robot's base frame is given by:

$$ {}^{0}_{n}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T\,{}^{4}_{5}T\,{}^{5}_{n}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_y \\ r_{31} & r_{32} & r_{33} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8} $$

where, r_{ij} represents the elements of the rotational part of the obtained transformation and d_x, d_y and d_z represent the translation components of this matrix. It is to be noted that the values of 'a's in the above matrices are negative in value, as seen earlier.

3.4 Inverse Kinematics

3.4.1 Position

For robot manipulators with 6 degrees of freedom and a non-intersecting wrist axes configuration, obtaining a geometric or closed-form solution of the inverse kinematics becomes close to impossible. Hence, for our ERA robot arm, the inverse position kinematics problem is solving [Eq. 3.8] given a desired transformation matrix of the end-effector with respect to the base frame, as shown on the right hand side of the equation. Expanding the values of each component in this transformation matrix where values of 'a's are negative and the components r_{11} to r_{33} and d_x, d_y, d_z are the desired

pose of the end-effector/tool for which the corresponding joint angles need to be computed,

$$r_{11} = c\theta_6(c\theta_5(c\theta_4(s\theta_1 s\theta_3 + c\theta_1 c\theta_3 s\theta_2) + s\theta_4(c\theta_3 s\theta_1 - c\theta_1 s\theta_2 s\theta_3)) + s\theta_5(c\theta_4(c\theta_3 s\theta_1 - c\theta_1 s\theta_2 s\theta_3) - s\theta_4(s\theta_1 s\theta_3 + c\theta_1 c\theta_3 s\theta_2))) + c\theta_1 c\theta_2 s\theta_6 \tag{3.9}$$

$$r_{12} = c\theta_1 c\theta_2 c\theta_6 - s\theta_6(c\theta_5(c\theta_4(s\theta_1 s\theta_3 + c\theta_1 c\theta_3 s\theta_2) + s\theta_4(c\theta_3 s\theta_1 - c\theta_1 s\theta_2 s\theta_3)) + s\theta_5(c\theta_4(c\theta_3 s\theta_1 - c\theta_1 s\theta_2 s\theta_3) - s\theta_4(s\theta_1 s\theta_3 + c\theta_1 c\theta_3 s\theta_2))) \tag{3.10}$$

$$r_{13} = c\theta_5(c\theta_4(c\theta_3 s\theta_1 - c\theta_1 s\theta_2 s\theta_3) - s\theta_4(s\theta_1 s\theta_3 + c\theta_1 c\theta_3 s\theta_2)) - s\theta_5(c\theta_4(s\theta_1 s\theta_3 + c\theta_1 c\theta_3 s\theta_2) + s\theta_4(c\theta_3 s\theta_1 - c\theta_1 s\theta_2 s\theta_3)) \tag{3.11}$$

$$r_{21} = c\theta_2 s\theta_1 s\theta_6 - c\theta_6(c\theta_5(c\theta_4(c\theta_1 s\theta_3 - c\theta_3 s\theta_1 s\theta_2) + s\theta_4(c\theta_1 c\theta_3 + s\theta_1 s\theta_2 s\theta_3)) + s\theta_5(c\theta_1 c\theta_3 c\theta_4 - c\theta_1 s\theta_3 s\theta_4 + c\theta_3 s\theta_1 s\theta_2 s\theta_4 + c\theta_4 s\theta_1 s\theta_2 s\theta_3)) \tag{3.12}$$

$$r_{22} = s\theta_6(c\theta_5(c\theta_4(c\theta_1 s\theta_3 - c\theta_3 s\theta_1 s\theta_2) + s\theta_4(c\theta_1 c\theta_3 + s\theta_1 s\theta_2 s\theta_3)) + s\theta_5(c\theta_1 c\theta_3 c\theta_4 - c\theta_1 s\theta_3 s\theta_4 + c\theta_3 s\theta_1 s\theta_2 s\theta_4 + c\theta_4 s\theta_1 s\theta_2 s\theta_3)) + c\theta_2 c\theta_6 s\theta_1 \tag{3.13}$$

$$r_{23} = s\theta_5(c\theta_4(c\theta_1 s\theta_3 - c\theta_3 c(\theta_2 - \frac{\pi}{2})s\theta_1) + s\theta_4(c\theta_1 c\theta_3 + c(\theta_2 - \frac{\pi}{2})s\theta_1 s\theta_3)) - c\theta_5(c\theta_4(c\theta_1 c\theta_3 + c(\theta_2 - \frac{\pi}{2})s\theta_1 s\theta_3) - s\theta_4(c\theta_1 s\theta_3 - c\theta_3 c(\theta_2 - \frac{\pi}{2})s\theta_1)) \tag{3.14}$$

$$r_{31} = s\theta_2 s\theta_6 - c\theta_6(c(\theta_2 + \theta_3 + \theta_4 + \theta_5)/2 + c(\theta_3 - \theta_2 + \theta_4 + \theta_5)/2) \tag{3.15}$$

$$r_{32} = c\theta_6 s\theta_2 + s\theta_6(c(\theta_2 + \theta_3 + \theta_4 + \theta_5)/2 + c(\theta_3 - \theta_2 + \theta_4 + \theta_5)/2) \tag{3.16}$$

$$r_{33} = s(\theta_3 - \theta_2 + \theta_4 + \theta_5)/2 + s(\theta_2 + \theta_3 + \theta_4 + \theta_5)/2 \tag{3.17}$$

$$\begin{aligned} d_x = {} & a_4 c(\theta_4)(s\theta_1 s\theta_3 + c\theta_1 c\theta_3 s\theta_2) + a4 s\theta_4 (c\theta_3 s\theta_1 - c\theta_1 s\theta_2 s\theta_3) \\ & + a6 c\theta_6 (c\theta_5 (c\theta_4 (s\theta_1 s\theta_3 + c\theta_1 c\theta_3 s\theta_2) + s\theta_4 (c\theta_3 s\theta_1 - c\theta_1 s\theta_2 s\theta_3)) \\ & \qquad + s\theta_5 (c\theta_4 (c\theta_3 s\theta_1 - c\theta_1 s\theta_2 s\theta_3) - s\theta_4 (s\theta_1 s\theta_3 + c\theta_1 c\theta_3 s\theta_2))) \\ & + a2 c\theta_1 s\theta_2 + a5 c\theta_5 (c\theta_4 (s\theta_1 s\theta_3 + c\theta_1 c\theta_3 s\theta_2) + s\theta_4 (c\theta_3 s\theta_1 - c\theta_1 s\theta_2 s\theta_3)) \\ & + a3 s\theta_1 s\theta_3 + a5 s\theta_5 (c\theta_4 (c\theta_3 s\theta_1 - c\theta_1 s\theta_2 s\theta_3) - s\theta_4 (s\theta_1 s\theta_3 + c\theta_1 c\theta_3 s\theta_2)) \\ & + a3 c\theta_1 c\theta_3 s\theta_2 + a6 c\theta_1 c\theta_2 s\theta_6 \end{aligned} \tag{3.18}$$

$$\begin{aligned} d_y = {} & a_2 s\theta_1 s\theta_2 - a_4 s\theta_4 (c\theta_1 c\theta_3 + s\theta_1 s\theta_2 s\theta_3) - a_3 c\theta_1 s\theta_3 \\ & - a_5 c\theta_5 (c\theta_4 (c\theta_1 s\theta_3 - c\theta_3 s\theta_1 s\theta_2) + s\theta_4 (c\theta_1 c\theta_3 + s\theta_1 s\theta_2 s\theta_3)) \\ & - a_6 c\theta_6 (c\theta_5 (c\theta_4 (c\theta_1 s\theta_3 - c\theta_3 s\theta_1 s\theta_2) + s\theta_4 (c\theta_1 c\theta_3 + s\theta_1 s\theta_2 s\theta_3)) \\ & \qquad + s\theta_5 (c\theta_1 c\theta_3 c\theta_4 - c\theta_1 s\theta_3 s\theta_4 + c\theta_3 s\theta_1 s\theta_2 s\theta_4 + c\theta_4 s\theta_1 s\theta_2 s\theta_3)) \\ & - a_4 c\theta_4 (c\theta_1 s\theta_3 - c\theta_3 s\theta_1 s\theta_2) \\ & - a_5 s\theta_5 (c\theta_1 c\theta_3 c\theta_4 - c\theta_1 s\theta_3 s\theta_4 + c\theta_3 s\theta_1 s\theta_2 s\theta_4 + c\theta_4 s\theta_1 s\theta_2 s\theta_3) \\ & + a_3 c\theta_3 s\theta_1 s\theta_2 + a_6 c\theta_2 s\theta_1 s\theta_6 \end{aligned} \tag{3.19}$$

$$\begin{aligned} d_z = {} & d_1 - a_2 c\theta_2 - a_3 c\theta_2 c\theta_3 + a_6 s\theta_2 s\theta_6 - a_4 c\theta_2 c\theta_3 c\theta_4 \\ & + a_4 c\theta_2 s\theta_3 s\theta_4 - a_5 c\theta_2 c\theta_3 c\theta_4 c\theta_5 + a_5 c\theta_2 c\theta_3 s\theta_4 s\theta_5 \\ & + a_5 c\theta_2 c\theta_4 s\theta_3 s\theta_5 + a_5 c\theta_2 c\theta_5 s\theta_3 s\theta_4 + a_6 c\theta_2 c\theta_3 c\theta_6 s\theta_4 s\theta_5 \\ & + a_6 c\theta_2 c\theta_4 c\theta_6 s\theta_3 s\theta_5 + a_6 c\theta_2 c\theta_5 c\theta_6 s\theta_3 s\theta_4 - a_6 c\theta_2 c\theta_3 c\theta_4 c\theta_5 c\theta_6 \end{aligned} \tag{3.20}$$

where, $c\theta_x = \cos x$ and $s\theta_x = \sin x$

The above 12 non-linear trigonometric equations are solved using the fsolve function on MATLAB using the Optimization Toolbox with an initial guess for the joint angles to obtain the resultant joint angles. In order to enforce joint limits on the obtained solution, a least-squares optimization can be performed with appropriate upper and lower limits on the joint variables.

3.4.2 Velocity

The inverse kinematics problem is transforming the end-effector Cartesian velocities (both angular and linear) into the corresponding joint space velocities. To achieve this, the 6x6 Jacobian matrix of the robot arm [Fig. 3.3] was setup using the Method-2 (partial differentiation of the translational component of the transformation matrix, ${}^0_nT$). The Z_{i-1} vectors were found from each ${}^0_iT$ matrix of each joint.

$$\dot{q} = J(q)^{-1}\dot{X} \tag{3.21}$$

where, q is the vector of all joint angles, J(q) is the Jacobian matrix and X is the vector of end-effector velocities.

Now, using $\dot{q}$, the joint angles q can be computed as follows, which are given as the desired joint trajectory for the manipulator:

$$q_{new} = q_{old} + \dot{q}\delta t \tag{3.22}$$

In our application, the cutter end-effector has to move in a straight line to cut through the insulation. For this, the velocity vector X contains a non-zero value in the appropriate linear velocity component of the vector (in our case 0.1 m/s in the Z direction). This is validated using ROS and Gazebo which is presented in the subsequent chapters.

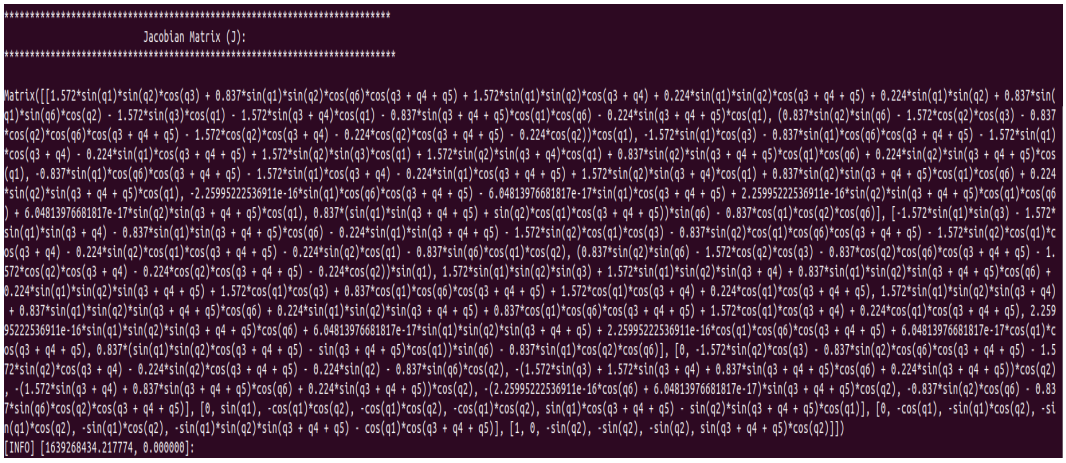

Figure 3.3: Generated Jacobian Matrix of the custom ERA robot

3.5 Kinematics Validation

The obtained forward kinematics transformation matrices are validated using a MATLAB script (*era_forward_kinematics.m*) for three cases namely: (a) when all joint angles are zero (home position) (b) when $\theta_2 = 90$ degrees and the other angles = 0 (c) when θ_1 and θ_4 are 90 degrees and the other angles are zero. [Fig. 3.4] shows the output of the code which gives satisfactory results when the corresponding geometric solution is used as reference.

Appendix A.1 contains the MATLAB script to validate ERA's obtained forward kinematics.

```
>> era_forward_kinematics
Case 1: All Joint Angles (rad) are 0
     0     0     0     0     0     0

Resultant transformation matrix between cutter tool blade and base frame:
    0.0000    1.0000         0   -0.0000
         0         0   -1.0000         0
   -1.0000    0.0000         0    5.0060
         0         0         0    1.0000

Case 2: q2 = 90, other angles = 0
         0    1.5708         0         0         0         0

Resultant transformation matrix between cutter tool blade and base frame:
    1.0000         0         0   -4.4290
         0         0   -1.0000         0
         0    1.0000         0    0.5770
         0         0         0    1.0000

Case 3: q1, q4 = 90, other angles = 0
    1.5708         0         0    1.5708         0         0

Resultant transformation matrix between cutter tool blade and base frame:
    1.0000    0.0000    0.0000   -2.6330
   -0.0000    1.0000   -0.0000    0.0000
   -0.0000    0.0000    1.0000    2.3730
         0         0         0    1.0000
```

Figure 3.4: Output of FK using Matlab

The inverse position kinematics (*era_inverse_kinematics.m*) of the robot is validated by solving the 12 non-linear trigonometric equations of the transformation matrix using MATLAB's fsolve function with an initial guess for the solution. [Fig. 3.5] shows the output of the code when the previously obtained X, Y & Z coordinates of the end-effector for all joint angles are zero are fed as input to this code, giving the right answer. The inverse velocity kinematics using the Jacobian matrix was not validated using MATLAB as this is done in ROS & Gazebo, discussed in the subsequent chapters.

Appendix A.2 contains the MATLAB script to validate ERA's inverse position kinematics.

```
>> era_inverse_kinematics
Input End-Effector positions:
         0         0    5.0060


Solving problem using fsolve.

Equation solved at initial point.

fsolve completed because the vector of function values at the initial
point is near zero as measured by the value of the function tolerance,
and the problem appears regular as measured by the gradient.

<stopping criteria details>
Resultant joint angles from IK:
     0
     0
     0
     0
     0
     0
```

Figure 3.5: Output of IK using fsolve Matlab script

Both the forward and inverse kinematics of the robot are validated and visualized using the Peter Corke MATLAB Robotics Toolbox[4] script (*satellite_servicer_kinematics_fk_ik.m*) for the previously mentioned three cases [Fig. 3.6]. The corresponding robot poses for these three cases are plotted as shown in [Fig. 3.7].

Appendix A.3 contains the MATLAB script implementation with Peter Corke Robotics Toolbox.

[4]Peter Corke MATLAB Toolbox link: `https://petercorke.com/toolboxes/robotics-toolbox/`

Figure 3.6: Output of FK & IK using Peter Corke Toolbox Matlab script

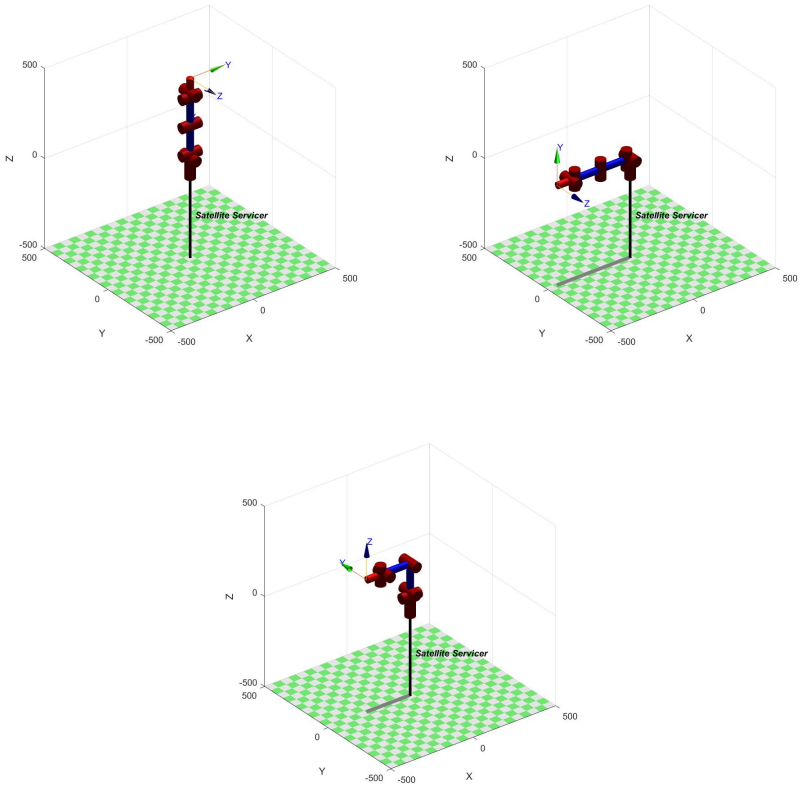

Figure 3.7: Robot plot: FK & IK (3 cases) validation using Matlab script

3.6 Workspace Study

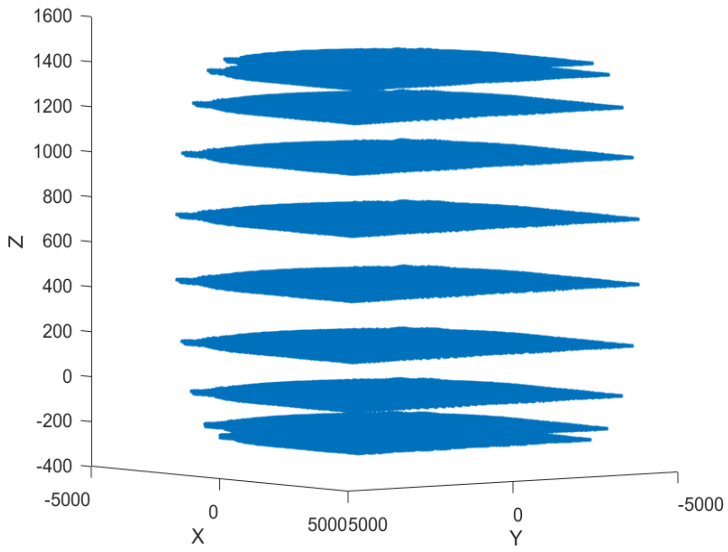

Figure 3.8: ERA Robot arm Workspace

The ERA robot arm has a total of 6 degrees of freedom which allows it to manipulate anywhere in 6 degrees of freedom space. To ascertain this, the workspace of the robot is plotted using MATLAB. Each joint angle is taken to be limited to the range $[\frac{-\pi}{2}, \frac{\pi}{2}]$. In observing the reachable workspace, the robot is assumed to be fixed on the ground with no satellite or obstacle around thus making it free to actuate to any point. [Fig. 3.8] shows a plot of the robot's reachable workspace plotted for 10 set of joint values. The curve represents a sphere thus showing the reachability of the manipulator. The maximum limits on each axis is approximately 5000 mm (or 5m) which is equal to 5006 mm, the designed maximum reachable length of the robot which is in accordance with the expectation. Below is the MATLAB script, *era_workspace.m*, used to generate this plot.

Appendix A.4 contains the MATLAB script to generate the above ERA robot workspace.

Chapter 4

Robot Control

Having developed the kinematic model required for simulation in the previous chapter, it is now important to setup a closed-loop controller for the joints of the robot arm to better track a desired reference signal. In this chapter, the *ros-control* [15] package of ROS and Gazebo is used for the implementation and validation of the controller.

Firstly, a <transmission> element of type "*SimpleTransmission*" and hardware interface of type "*EffortJointInterface*" were added to each joint in the robot's URDF file to link the joint with an actuator. To be able to perform position kinematics of the robot as obtained earlier, a simple PID controller was defined for each joint of the robot arm to track a given desired joint position/angle. The input to the controller is an angle set point while the PID controller outputs an actuating signal to the robot plant in the form of an input force/torque at that joint to actuate it to the desired setting. The type used is "*effort_controller/JointPositionController*".

Setting up the above ROS and Gazebo plugin gives access to a new ROS topic of the form "robot-name/controller-name/command/data" which contains a ROS message of type "std_msgs/Float64" to command the desired position (in radians) to the corresponding joint. In order to be able to visualize the performance of this new PID controller and to tune its gains (K_p, K_i, K_d), ROS's plugin-based user interface RQT GUI was used to define an input joint position trajectory and observe the controller's response to it. The ROS topic "robot-name/controller-name/state/process_value" was used to track the actual position of the actuator being controlled. The goal is to minimize the error between the actual and desired joint positions while accounting for other measures of the time-response such as overshoot, oscillations, rise time and steady state error. [16]

Using the 'Dynamic Reconfigure' option in RQT with an input position trajectory of a sine wave defined as:

$$\theta_{des} = sin(\frac{i}{rate * speed}) * diff + offset \tag{4.1}$$

where,

- i - simulation time in seconds (in RQT)
- rate - frequency of the evaluation of this equation (set to 50Hz)
- speed - Speed of actuating the joint (set to 1 for slow speed)
- diff = $\frac{upper\_limit - lower\_limit}{2}$
- offset = upper_limit - diff
- upper_limit & lower_limit - joints angle limits (here, $\pi/2$ and -$\pi/2$ radians respectively.)

The obtained plots of the actual and desired joint position trajectory input (for joints 3 to 6) from Eqn 4.1 for the tuned and untuned pair of PID gains are given below. There is a clear improvement in the performance of the system due to PID gain tuning. The general process of tuning the controller was to initially take K_i and K_d zero and a non-zero value of K_p. This K_p was increased until a point where the rise time and settling time were satisfactory with little overshoot and oscillations. To counter these oscillations, the K_d gain as increased keeping K_p constant until a satisfactory oscillatory response was obtained. Finally, any steady state error in the system was eliminated/minimized by increasing the K_i gain. The final tuned values of PID gains for each joint are reported in [Table 4.1].

It is to be noted that the tuning of these gains was done with the existence of gravity (g = -9.8m/s^2) whereas the actual simulation of the robot is done in zero/low gravity in Gazebo. Also, due to the nature of the desired input angle, sometimes the robot would fall on the ground and hence regions of saturation despite an input in the subsequent plots indicates the arm was not able to move despite an input due to an obstruction.

Table 4.1: PID Gains for custom ERA robot joints

Joint	K_p	K_i	K_d
Joint 1	1000.0	120.0	220.0
Joint 2	100000.0	520.0	1020.0
Joint 3	8000.0	220.0	320.0
Joint 4	8000.0	220.0	320.0
Joint 5	8000.0	220.0	320.0
Joint 6	4000.0	40.0	100.0

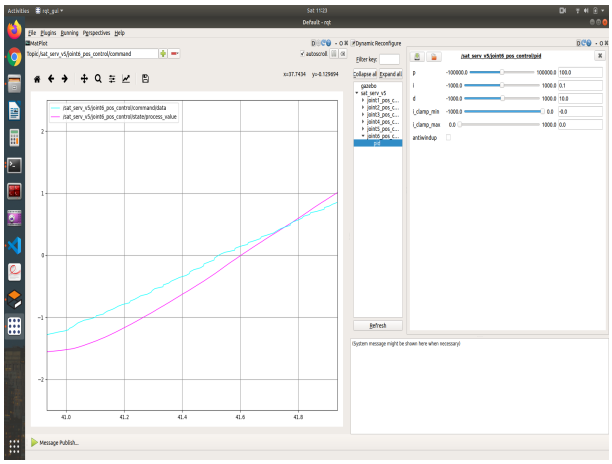

Figure 4.1: Joint 6 Untuned PID gains plot

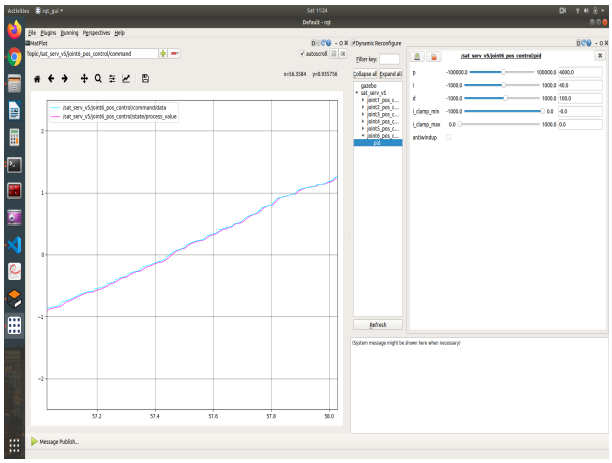

Figure 4.2: Joint 6 Tuned PID gains plot

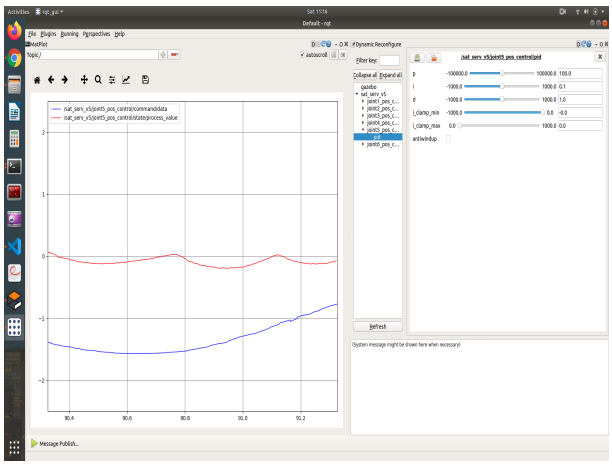

Figure 4.3: Joint 5 Untuned PID gains plot

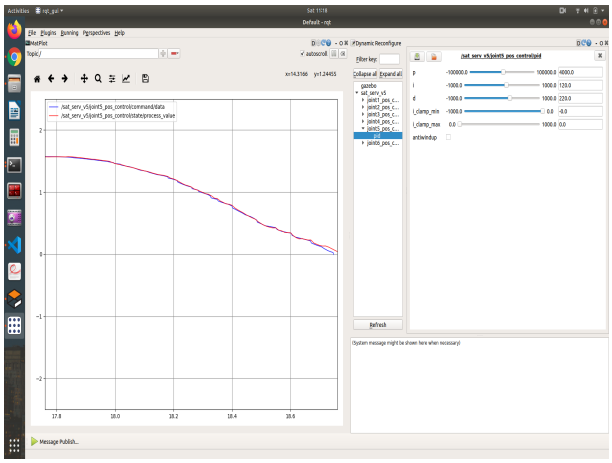

Figure 4.4: Joint 5 Tuned PID gains plot

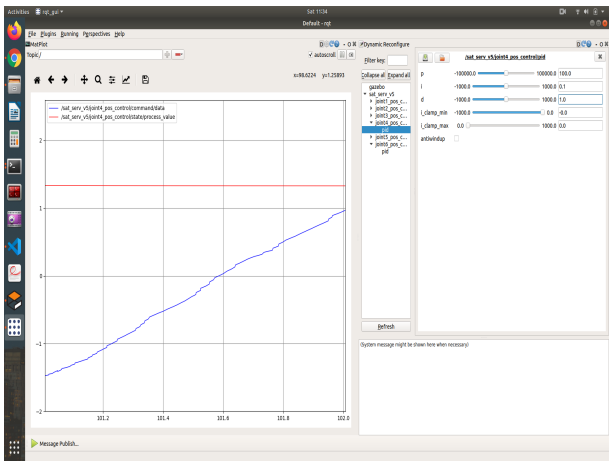

Figure 4.5: Joint 4 Untuned PID gains plot

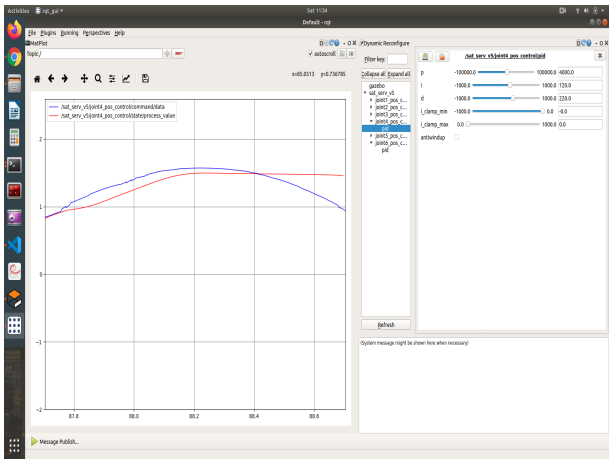

Figure 4.6: Joint 4 Tuned PID gains plot

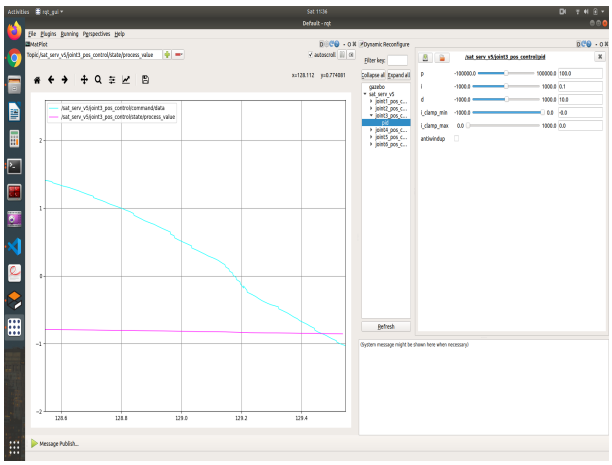

Figure 4.7: Joint 3 Untuned PID gains plot

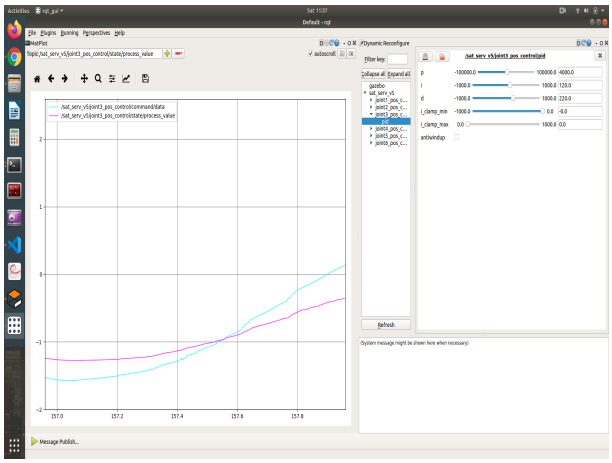

Figure 4.8: Joint 3 Tuned PID gains plot

Chapter 5

Simulation

5.1 Gazebo Visualization

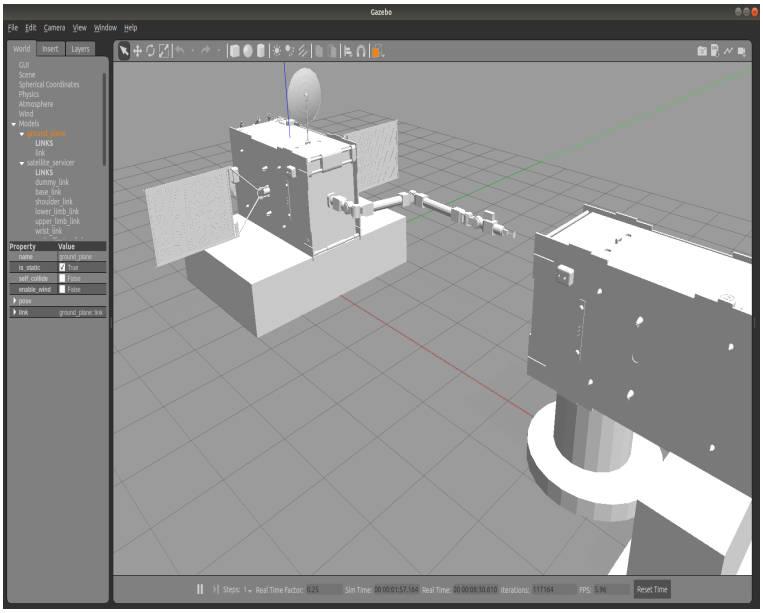

Figure 5.1: Gazebo Visualization

In order to simulate the ERA robotic arm plus satellite system, Gazebo is used to represent the physics. In the Gazebo world [Fig. 5.1], the servicing satellite and robot arm assembly is placed rigidly on the ground at the origin of the Gazebo world. The satellite to be serviced, also called broken satellite, is placed at an offset along the X axis from the servicing assembly. The entire assembly is fixed to the ground to emulate the case when both the satellites have zero relative velocity between them in space. The gravity parameter under the Physics tab in Gazebo is set to -2.5 su ch that it is just enough to represent low gravity while not being too low to cause unwarranted motions.

The tele-operation demonstration of the system incorporates time delay between an input ROS topic and the output ROS topic to recreate the round-trip telemetry delay between Earth and Space. This is akin to a human operator manually controlling the satellite and attempting to service the satellite from ground. The cutting demonstration does not incorporate time delay as ideally this script would be running on the robot arm's computer thereby eliminating the need for a human operator so as to not incur any time delay and potential mishaps due to poor visualization interfaces.

5.2 RViz Visualization

As mentioned previously, visualization interfaces play a key role in the ability to see the cutting (or servicing in general) in action. To facilitate this, two cameras have been placed [Fig. 5.2] on the end-effector at a 180 degrees difference such that one camera looks at the cutter blade to give visual feedback of the cutting process back to the ground station while the other camera looks at the satellite to which the arm is mounted to give an idea of the position and orientation of the robot arm to the ground operator.

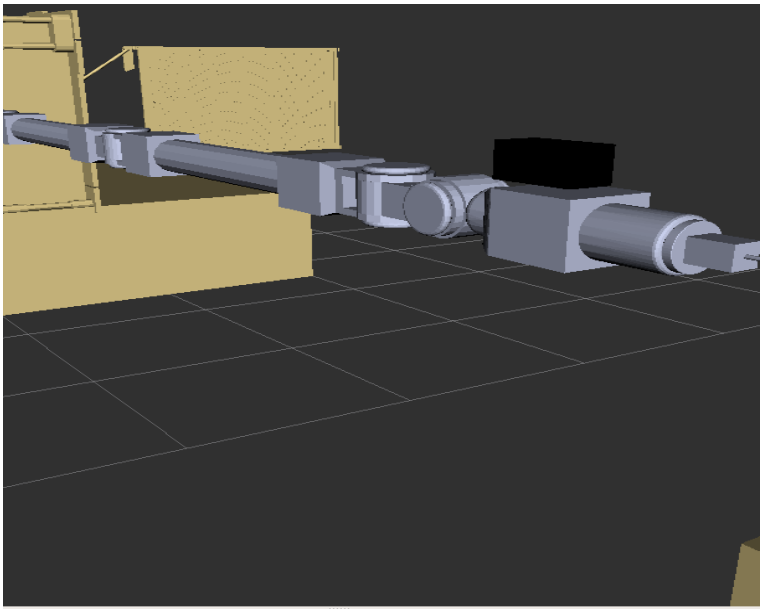

Figure 5.2: Cameras (black) mounted on the ERA arm

[Fig. 5.3] shows the entire satellite servicing assembly in RViz with camera views from both the mounted cameras on the bottom left and right sides. Also, visible is the GUI of the joint_state_publisher node that is used to visualize the changing of joint angles in RViz. This GUI is connected to the Gazebo environment as an implementation of the teleoperation activity discussed previously. The tf package is also visualized in RViz to check for the proper assigning of co-ordinate frames on the robot arm in accordance to the previously discussed D-H convention.

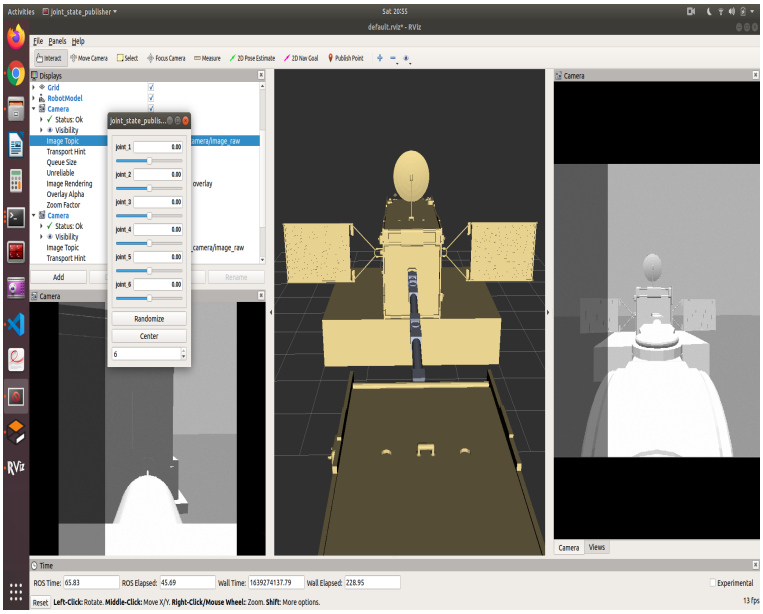

Figure 5.3: RViz Visualization

5.3 ROS Computation Graph

The below figures [Fig. 5.4 & 5.5] display the ROS computation graphs corresponding to running the cutting and teleoperation demos in Gazebo and RViz. The different ROS nodes are marked in oval shape whereas the ROS topics are shown in rectangles. The *rqt_graph* tool in ROS provides a GUI plugin to visualise this graph. Alternatively, the *rosgraph* command provides a command line version of rqt_graph displaying periodic graph information in text format [17]. It is to be noted that the presence of a 'delayed' node

in the teleop rqt graph which takes a ROS topic as input and outputs a delayed version of the same ROS topic, used to establish telemetry delays.

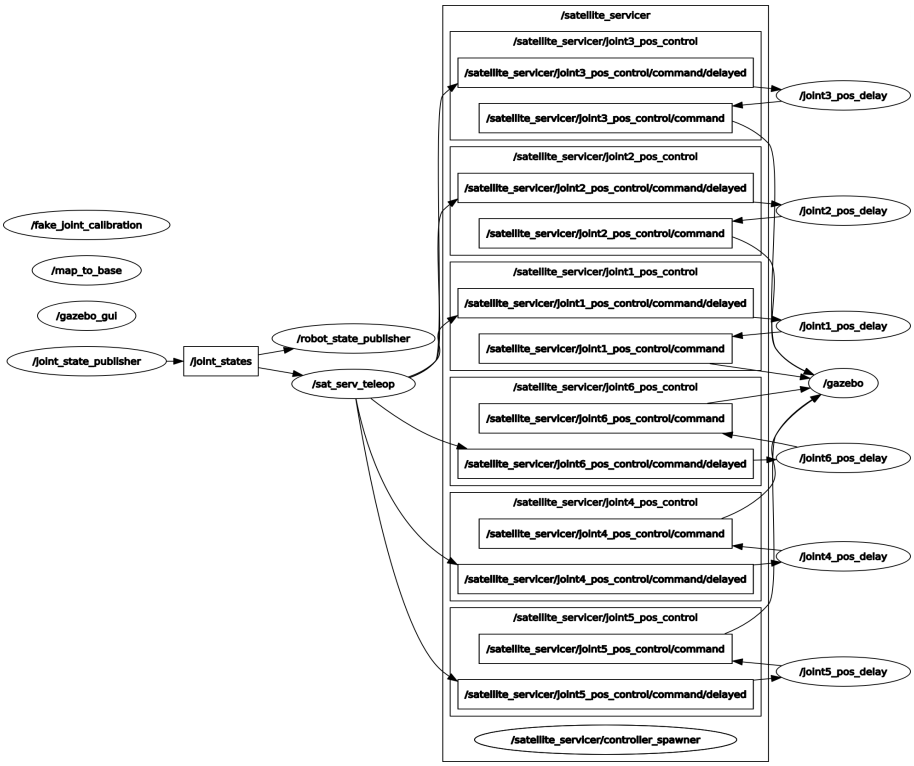

Figure 5.4: RQT Graph for Teleop demo

The links to the demo videos of running teleoperation and cutting functions of the servicing satellite are in the following Google Drive link: `https://drive.google.com/drive/folders/1vJnj-hokhnRyL8YHneCsOHNl8h1R-Bu3?usp=sharing`

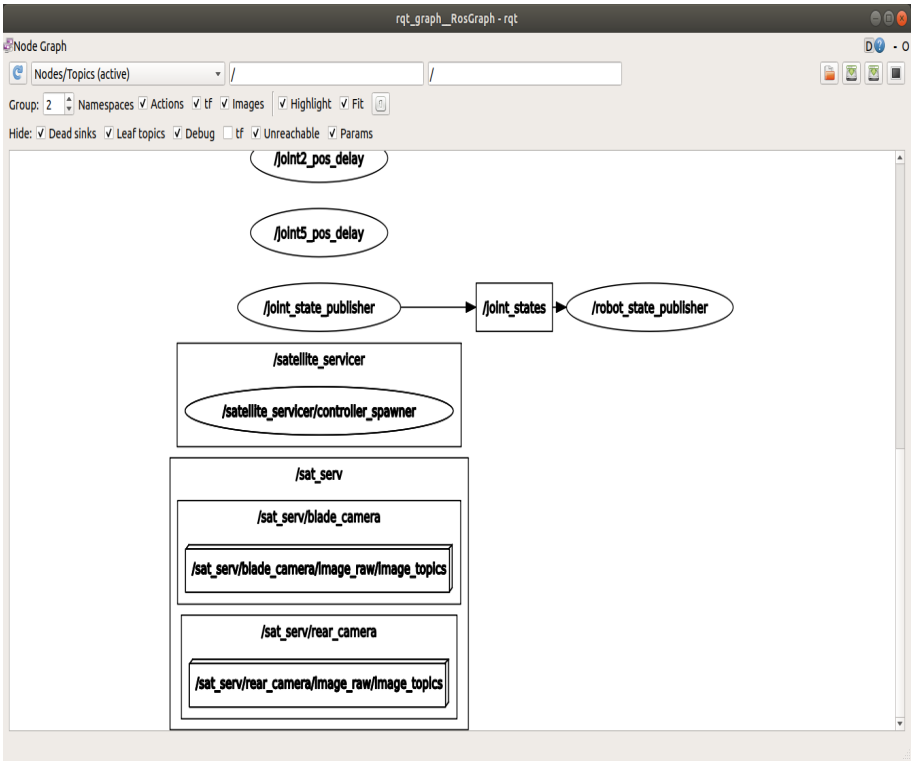

Figure 5.5: RQT Graph for Cutting demo

Chapter 6

Challenges Faced & Lessons Learned

- Simulating the arm in very low gravity means a small impulse due to joint motion shall put the entire assembly into undesirable motion. Also, a base frame had to be mounted to both service and receiver satellites as they tend to elevate in the simulation due to low gravity.
- Spawning a custom designed manipulator without proper knowledge of joint limits and gravity compensation led to the requirement of high PID gains tuning. By making rapid experimentation on expected PID gains, we were able to optimize the gain values to the best possibility.
- Developing a closed form solution for inverse kinematics of 6 DoF robot manipulators with non-intersecting wrist axes is complicated. All these parameter were taken into consideration while designing the ERA in CAD.
- Improper definition of joint axes and coordinate frames to each link in joint in the CAD assembly made our robot to 'explode' in the gazebo simulation. Assigning of coordinate frames in local view instead of a global approach in the assembly section of CAD modelling led to this problem.
- The weight of the robot is very high and coupled with the long length of the limb, trajectory tracking often has errors as this entire weight is handled by one or two joints in the base.
- Computing the symbolic representation of the Jacobian matrix of the robot arm and further computing its inverse in the inverse velocity kinematics equations is computationally intensive and takes a lot of time. Wherever possible, incorporate already coded up equations such that the run-time is optimized.

- Using an already existing satellite CAD model from the internet made it initially difficult for our simulations as this model contained numerous meshes thereby eating graphics and system memory, as the CAD model was highly detailed. Certain parts were trimmed out to simplify the model.

Chapter 7

Conclusions & Future Work

The developed prototype of a servicing satellite is a hybrid model inspired from various space missions and autonomous systems including the ISS, NASA and ERA respectively. This model can reduce the cost of satellite service dramatically and also avoid human intervention thereby saving lives. The modeled and simulated tele-robotic arm can be used as a pragmatic solution to clear space debris, keeping our cosmos available for future missions. The arm has been developed to closely resemble the original ERA arm with the exception in the mass and length down scaling. It is seen to effectively manipulate the various degrees of freedom in outer space and cut through the target insulation cover of the satellite to be serviced.

The prototype can be extended to implement a part/tool replacement box on the host satellite from which the robot arm can replace the tool on the end-effector based on the servicing operation to be performed. A secondary robot arm can be mounted onto the system to assist the first robot in holding the target satellite steadily in place, interchange of end-effector tool depending on the required application and to extend the reachability of the servicing satellite into regions where the first arm cannot reach. Also by using the force analysis on the end-effector, the force required to sturdily grab the satellite to be serviced can be accurately determined. In order to better assist human operators from the mission control center on Earth, additional visualization and sensing technologies can be integrated into the servicing satellite. The robot arm can also be made smart to decide between tasks it can autonomously do and other activities where a human intervention is required, to better manage the round-trip communication telemetry delays between Earth and Space.

Bibliography

[1] W. Pryor, B. P. Vagvolgyi, W. J. Gallagher, A. Deguet, S. Leonard, L. L. Whitcomb, and P. Kazanzides, "Experimental evaluation of teleoperation interfaces for cutting of satellite insulation," in IEEE Intl. Conf. on Robotics and Automation (ICRA), May 2019, pp. 4775–4781

[2] About – Hubble Space Telescope History Timeline, NASA [Online], Available: https://www.nasa.gov/content/goddard/hubble-history-timeline

[3] Hubble Space Telescope Servicing Mission 3A – Cost to Taxpayers, NASA [Online], Available: https://asd.gsfc.nasa.gov/archive/sm3a/downloads/sm3a_fact_sheets/cost-to-taxpayers.pdf

[4] Orbital Space Debris and Human Spacecraft, NASA [Online] https://www.nasa.gov/mission_pages/station/news/orbital_debris.html

[5] On-Orbit Satellite Servicing Study, NASA SSPD Project Report [Online]. Available: https://nexis.gsfc.nasa.gov/images/NASA_Satellite%20Servicing_Project_Report_0511.pdf

[6] Paul, R. P. "Robot manipulators : mathematics, programming, and control : the computer control of robot manipulators", Cambridge, Mass.: MIT Press (1984).

[7] Denavit, J. & Hartenberg, R. S. (1955), 'A kinematic notation for lower-pair mechanisms based on matrices', Trans.ASME E, Journal of Applied Mechanics 22 , 215-221.

[8] Spong, M. W.; Hutchinson, S.; Vidyasagar, "M. Robot Modeling and Control"; John Wiley & Sons: Hoboken, NJ, 2006.

[9] Zamanzadeh, A., Ahmadi, H. (2021). Inverse kinematic of European Robotic Arm based on a new geometrical approach. AUT Journal of Mechanical Engineering, 5(1), 13-48. doi: 10.22060/ajme.2020.17642.5866

[10] European Robotic Arm (ERA) The International Space Station's Latest Upgrade, [Online]: https://esamultimedia.esa.int/docs/science/ERA_brochure_EN.pdf

[11] H.J. Cruijssen; M. Ellenbroek; M. Henderson; H. Petersen; P. Verzijden; M. Visser (May 2014). "42nd Aerospace Mechanism Symposium: The European Robotic Arm: A High-Performance Mechanism Finally on its way to Space" (PDF). NASA. pp. 321–323, 329.

[12] European Robotic Arm Factsheet, [Online]: http://wsn.spaceflight.esa.int/docs/Factsheets/7%20ERA%20LR.pdf

[13] https://grabcad.com/library/satellite-turksat-5a-1

[14] https://www.mathworks.com/help/optim/ug/optim.problemdef.equationproblem.html

[15] https://wiki.ros.org/ros_control

[16] http://gazebosim.org/tutorials/?tut=ros_control

[17] ROS Documentation - http://wiki.ros.org/

Appendix A

Codes

A.1 Forward Kinematics validation MATLAB script

```
% era_forward_kinematics.m

a2 = -0.224; a3 = -1.572; a4 = -1.572; a5 = -0.224; a6 = -(0.577 +
    0.260);

disp("Case 1: All Joint Angles (rad) are 0 ")
theta1 = 0 * pi/180;
theta2 = 0 * pi/180;
theta3 = 0 * pi/180;
theta4 = 0 * pi/180;
theta5 = 0 * pi/180;
theta6 = 0 * pi/180;
theta = [theta1, theta2, theta3, theta4, theta5, theta6];
disp(theta)
% T_01 = t_matrix(a,d,theta,alpha)
T_01 =
    [[cos(theta1),0,sin(theta1),0];[sin(theta1),0,-cos(theta1),0];[0,1,0,d1];[0,0,0,1]];

T_12 =
    [[cos(theta2-(pi/2)),0,sin(theta2-(pi/2)),a2*cos(theta2-(pi/2))];[sin(theta2-(pi/2)),0,-c

T_23 =
    [[cos(theta3),-sin(theta3),0,a3*cos(theta3)];[sin(theta3),cos(theta3),0,a3*sin(theta3)];[

T_34 =
    [[cos(theta4),-sin(theta4),0,a4*cos(theta4)];[sin(theta4),cos(theta4),0,a4*sin(theta4)];[

T_45 =
    [[cos(theta5),0,-sin(theta5),a5*cos(theta5)];[sin(theta5),0,cos(theta5),a5*sin(theta5)];[
```

```
T_5n =
    [[cos(theta6),-sin(theta6),0,a6*cos(theta6)];[sin(theta6),cos(theta6),0,a6*sin(theta6)];[

T_0n = T_01 * T_12 * T_23 * T_34 * T_45 * T_5n;

disp("Resultant transformation matrix between cutter tool blade and
    base frame: ")
disp(T_0n)

disp("Case 2: q2 = 90, other angles = 0")
theta1 = 0 * pi/180;
theta2 = 90 * pi/180;
theta3 = 0 * pi/180;
theta4 = 0 * pi/180;
theta5 = 0 * pi/180;
theta6 = 0 * pi/180;
theta = [theta1, theta2, theta3, theta4, theta5, theta6];
disp(theta)
% T_01 = t_matrix(a,d,theta,alpha)
T_01 =
    [[cos(theta1),0,sin(theta1),0];[sin(theta1),0,-cos(theta1),0];[0,1,0,d1];[0,0,0,1]];

T_12 =
    [[cos(theta2-(pi/2)),0,sin(theta2-(pi/2)),a2*cos(theta2-(pi/2))];[sin(theta2-(pi/2)),0,-c

T_23 =
    [[cos(theta3),-sin(theta3),0,a3*cos(theta3)];[sin(theta3),cos(theta3),0,a3*sin(theta3)];[

T_34 =
    [[cos(theta4),-sin(theta4),0,a4*cos(theta4)];[sin(theta4),cos(theta4),0,a4*sin(theta4)];[

T_45 =
    [[cos(theta5),0,-sin(theta5),a5*cos(theta5)];[sin(theta5),0,cos(theta5),a5*sin(theta5)];[

T_5n =
    [[cos(theta6),-sin(theta6),0,a6*cos(theta6)];[sin(theta6),cos(theta6),0,a6*sin(theta6)];[

T_0n = T_01 * T_12 * T_23 * T_34 * T_45 * T_5n;

disp("Resultant transformation matrix between cutter tool blade and
    base frame: ")
disp(T_0n)

disp("Case 3: q1, q4 = 90, other angles = 0")
theta1 = 90 * pi/180;
```

```
theta2 = 0 * pi/180;
theta3 = 0 * pi/180;
theta4 = 90 * pi/180;
theta5 = 0 * pi/180;
theta6 = 0 * pi/180;
theta = [theta1, theta2, theta3, theta4, theta5, theta6];
disp(theta)
% T_01 = t_matrix(a,d,theta,alpha)
T_01 =
    [[cos(theta1),0,sin(theta1),0];[sin(theta1),0,-cos(theta1),0];[0,1,0,d1];[0,0,0,1]];

T_12 =
    [[cos(theta2-(pi/2)),0,sin(theta2-(pi/2)),a2*cos(theta2-(pi/2))];[sin(theta2-(pi/2)),0,-c

T_23 =
    [[cos(theta3),-sin(theta3),0,a3*cos(theta3)];[sin(theta3),cos(theta3),0,a3*sin(theta3)];[

T_34 =
    [[cos(theta4),-sin(theta4),0,a4*cos(theta4)];[sin(theta4),cos(theta4),0,a4*sin(theta4)];[

T_45 =
    [[cos(theta5),0,-sin(theta5),a5*cos(theta5)];[sin(theta5),0,cos(theta5),a5*sin(theta5)];[

T_5n =
    [[cos(theta6),-sin(theta6),0,a6*cos(theta6)];[sin(theta6),cos(theta6),0,a6*sin(theta6)];[

T_0n = T_01 * T_12 * T_23 * T_34 * T_45 * T_5n;

disp("Resultant transformation matrix between cutter tool blade and
    base frame: ")
disp(T_0n)
```

A.2 Inverse Kinematics validation MATLAB script [14]

```
% era_inverse_kinematics.m
% Computing the inverse kinematics using the Optimization Toolbox
d1 = 0.577;
a2 = -0.224; a3 = -1.572; a4 = -1.572; a5 = -0.224; a6 = -(0.577 +
    0.260);

theta = optimvar('theta',6);
r11 = 0; r12 = 1; r13 = 0;
r21 = 0; r22 = 0; r23 = -1;
r31 = -1; r32 = 0; r33 = 0.0;
disp("Input End-Effector positions: ")
dx = 0.0; dy = 0.0; dz = 5.006;
p = [dx,dy,dz];
disp(p)

e1 =
    cos(theta(6))*(cos(theta(5))*(cos(theta(4))*(sin(theta(1))*sin(theta(3))
    + cos(theta(1))*cos(theta(3))*sin(theta(2))) +
    sin(theta(4))*(cos(theta(3))*sin(theta(1)) -
    cos(theta(1))*sin(theta(2))*sin(theta(3)))) +
    sin(theta(5))*(cos(theta(4))*(cos(theta(3))*sin(theta(1)) -
    cos(theta(1))*sin(theta(2))*sin(theta(3))) -
    sin(theta(4))*(sin(theta(1))*sin(theta(3)) +
    cos(theta(1))*cos(theta(3))*sin(theta(2))))) +
    cos(theta(1))*cos(theta(2))*sin(theta(6)) == r11;
e2 = cos(theta(1))*cos(theta(2))*cos(theta(6)) -
    sin(theta(6))*(cos(theta(5))*(cos(theta(4))*(sin(theta(1))*sin(theta(3))
    + cos(theta(1))*cos(theta(3))*sin(theta(2))) +
    sin(theta(4))*(cos(theta(3))*sin(theta(1)) -
    cos(theta(1))*sin(theta(2))*sin(theta(3)))) +
    sin(theta(5))*(cos(theta(4))*(cos(theta(3))*sin(theta(1)) -
    cos(theta(1))*sin(theta(2))*sin(theta(3))) -
    sin(theta(4))*(sin(theta(1))*sin(theta(3)) +
    cos(theta(1))*cos(theta(3))*sin(theta(2))))) == r12;
e3 = cos(theta(5))*(cos(theta(4))*(cos(theta(3))*sin(theta(1)) -
    cos(theta(1))*sin(theta(2))*sin(theta(3))) -
    sin(theta(4))*(sin(theta(1))*sin(theta(3)) +
    cos(theta(1))*cos(theta(3))*sin(theta(2)))) -
    sin(theta(5))*(cos(theta(4))*(sin(theta(1))*sin(theta(3)) +
    cos(theta(1))*cos(theta(3))*sin(theta(2))) +
    sin(theta(4))*(cos(theta(3))*sin(theta(1)) -
```

```
    cos(theta(1))*sin(theta(2))*sin(theta(3)))) == r13;
e4 = cos(theta(2))*sin(theta(1))*sin(theta(6)) -
    cos(theta(6))*(cos(theta(5))*(cos(theta(4))*(cos(theta(1))*sin(theta(3))
    - cos(theta(3))*sin(theta(1))*sin(theta(2))) +
    sin(theta(4))*(cos(theta(1))*cos(theta(3)) +
    sin(theta(1))*sin(theta(2))*sin(theta(3)))) +
    sin(theta(5))*(cos(theta(1))*cos(theta(3))*cos(theta(4)) -
    cos(theta(1))*sin(theta(3))*sin(theta(4)) +
    cos(theta(3))*sin(theta(1))*sin(theta(2))*sin(theta(4)) +
    cos(theta(4))*sin(theta(1))*sin(theta(2))*sin(theta(3)))) ==
    r21;
e5 =
    sin(theta(6))*(cos(theta(5))*(cos(theta(4))*(cos(theta(1))*sin(theta(3))
    - cos(theta(3))*sin(theta(1))*sin(theta(2))) +
    sin(theta(4))*(cos(theta(1))*cos(theta(3)) +
    sin(theta(1))*sin(theta(2))*sin(theta(3)))) +
    sin(theta(5))*(cos(theta(1))*cos(theta(3))*cos(theta(4)) -
    cos(theta(1))*sin(theta(3))*sin(theta(4)) +
    cos(theta(3))*sin(theta(1))*sin(theta(2))*sin(theta(4)) +
    cos(theta(4))*sin(theta(1))*sin(theta(2))*sin(theta(3)))) +
    cos(theta(2))*cos(theta(6))*sin(theta(1)) == r22;
e6 = sin(theta(5))*(cos(theta(4))*(cos(theta(1))*sin(theta(3)) -
    cos(theta(3))*cos(theta(2) - pi/2)*sin(theta(1))) +
    sin(theta(4))*(cos(theta(1))*cos(theta(3)) + cos(theta(2) -
    pi/2)*sin(theta(1))*sin(theta(3)))) -
    cos(theta(5))*(cos(theta(4))*(cos(theta(1))*cos(theta(3)) +
    cos(theta(2) - pi/2)*sin(theta(1))*sin(theta(3))) -
    sin(theta(4))*(cos(theta(1))*sin(theta(3)) -
    cos(theta(3))*cos(theta(2) - pi/2)*sin(theta(1)))) == r23;
e7 = sin(theta(2))*sin(theta(6)) - cos(theta(6))*(cos(theta(2) +
    theta(3) + theta(4) + theta(5))/2 + cos(theta(3) - theta(2) +
    theta(4) + theta(5))/2) == r31;
e8 = cos(theta(6))*sin(theta(2)) + sin(theta(6))*(cos(theta(2) +
    theta(3) + theta(4) + theta(5))/2 + cos(theta(3) - theta(2) +
    theta(4) + theta(5))/2) == r32;
e9 = sin(theta(3) - theta(2) + theta(4) + theta(5))/2 +
    sin(theta(2) + theta(3) + theta(4) + theta(5))/2 == r33;

e10 = a4*cos(theta(4))*(sin(theta(1))*sin(theta(3)) +
    cos(theta(1))*cos(theta(3))*sin(theta(2))) +
    a4*sin(theta(4))*(cos(theta(3))*sin(theta(1)) -
    cos(theta(1))*sin(theta(2))*sin(theta(3))) +
    a6*cos(theta(6))*(cos(theta(5))*(cos(theta(4))*(sin(theta(1))*sin(theta(3))
    + cos(theta(1))*cos(theta(3))*sin(theta(2))) +
    sin(theta(4))*(cos(theta(3))*sin(theta(1)) -
```

```
    cos(theta(1))*sin(theta(2))*sin(theta(3)))) +
    sin(theta(5))*(cos(theta(4))*(cos(theta(3))*sin(theta(1)) -
    cos(theta(1))*sin(theta(2))*sin(theta(3))) -
    sin(theta(4))*(sin(theta(1))*sin(theta(3)) +
    cos(theta(1))*cos(theta(3))*sin(theta(2))))) +
    a2*cos(theta(1))*sin(theta(2)) +
    a5*cos(theta(5))*(cos(theta(4))*(sin(theta(1))*sin(theta(3)) +
    cos(theta(1))*cos(theta(3))*sin(theta(2))) +
    sin(theta(4))*(cos(theta(3))*sin(theta(1)) -
    cos(theta(1))*sin(theta(2))*sin(theta(3)))) +
    a3*sin(theta(1))*sin(theta(3)) +
    a5*sin(theta(5))*(cos(theta(4))*(cos(theta(3))*sin(theta(1)) -
    cos(theta(1))*sin(theta(2))*sin(theta(3))) -
    sin(theta(4))*(sin(theta(1))*sin(theta(3)) +
    cos(theta(1))*cos(theta(3))*sin(theta(2)))) +
    a3*cos(theta(1))*cos(theta(3))*sin(theta(2)) +
    a6*cos(theta(1))*cos(theta(2))*sin(theta(6)) == dx;
e11 = 2*sin(theta(1))*sin(theta(2)) -
    a4*sin(theta(4))*(cos(theta(1))*cos(theta(3)) +
    sin(theta(1))*sin(theta(2))*sin(theta(3))) -
    a3*cos(theta(1))*sin(theta(3)) -
    a5*cos(theta(5))*(cos(theta(4))*(cos(theta(1))*sin(theta(3)) -
    cos(theta(3))*sin(theta(1))*sin(theta(2))) +
    sin(theta(4))*(cos(theta(1))*cos(theta(3)) +
    sin(theta(1))*sin(theta(2))*sin(theta(3)))) -
    a6*cos(theta(6))*(cos(theta(5))*(cos(theta(4))*(cos(theta(1))*sin(theta(3))
    - cos(theta(3))*sin(theta(1))*sin(theta(2))) +
    sin(theta(4))*(cos(theta(1))*cos(theta(3)) +
    sin(theta(1))*sin(theta(2))*sin(theta(3)))) +
    sin(theta(5))*(cos(theta(1))*cos(theta(3))*cos(theta(4)) -
    cos(theta(1))*sin(theta(3))*sin(theta(4)) +
    cos(theta(3))*sin(theta(1))*sin(theta(2))*sin(theta(4)) +
    cos(theta(4))*sin(theta(1))*sin(theta(2))*sin(theta(3)))) -
    a4*cos(theta(4))*(cos(theta(1))*sin(theta(3)) -
    cos(theta(3))*sin(theta(1))*sin(theta(2))) -
    a5*sin(theta(5))*(cos(theta(1))*cos(theta(3))*cos(theta(4)) -
    cos(theta(1))*sin(theta(3))*sin(theta(4)) +
    cos(theta(3))*sin(theta(1))*sin(theta(2))*sin(theta(4)) +
    cos(theta(4))*sin(theta(1))*sin(theta(2))*sin(theta(3))) +
    a3*cos(theta(3))*sin(theta(1))*sin(theta(2)) +
    a6*cos(theta(2))*sin(theta(1))*sin(theta(6)) == dy;
e12 = d1 - a2*cos(theta(2)) - a3*cos(theta(2))*cos(theta(3)) +
    a6*sin(theta(2))*sin(theta(6)) -
    a4*cos(theta(2))*cos(theta(3))*cos(theta(4)) +
    a4*cos(theta(2))*sin(theta(3))*sin(theta(4)) -
```

```
    a5*cos(theta(2))*cos(theta(3))*cos(theta(4))*cos(theta(5)) +
    a5*cos(theta(2))*cos(theta(3))*sin(theta(4))*sin(theta(5)) +
    a5*cos(theta(2))*cos(theta(4))*sin(theta(3))*sin(theta(5)) +
    a5*cos(theta(2))*cos(theta(5))*sin(theta(3))*sin(theta(4)) +
    a6*cos(theta(2))*cos(theta(3))*cos(theta(6))*sin(theta(4))*sin(theta(5))
    +
    a6*cos(theta(2))*cos(theta(4))*cos(theta(6))*sin(theta(3))*sin(theta(5))
    +
    a6*cos(theta(2))*cos(theta(5))*cos(theta(6))*sin(theta(3))*sin(theta(4))
    -
    a6*cos(theta(2))*cos(theta(3))*cos(theta(4))*cos(theta(5))*cos(theta(6))
    == dz;

prob = eqnproblem;
prob.Equations.eq1 = e1;
prob.Equations.eq2 = e2;
prob.Equations.eq3 = e3;
prob.Equations.eq4 = e4;
prob.Equations.eq5 = e5;
prob.Equations.eq6 = e6;
prob.Equations.eq7 = e7;
prob.Equations.eq8 = e8;
prob.Equations.eq9 = e9;
prob.Equations.eq10 = e10;
prob.Equations.eq11 = e11;
prob.Equations.eq12 = e12;

x0.theta = [0 0 0 0 0 0];
[sol,fval,exitflag] = solve(prob,x0);
disp("Resultant joint angles from IK: ")
disp(sol.theta)
```

A.3 Forward & Inverse Kinematics validation using Peter Corke MATLAB Robotics Toolbox

```
% satellite_servicer_fk_ik.m
% FK and IK validation of ERA Satellite Servicing robot

% Link offsets (d) [m]
d1 = 0.577;

% Link lengths (a) [m]
a2 = -0.224; a3 = -1.572; a4 = -1.572; a5 = -0.224; a6 = -0.577 -
    0.260;

% D-H Table
% L(i)= Link ( [theta  d     a     alpha  ])
  L(1)= Link ( [0,     d1,   0,    pi/2   ]);
  L(2)= Link ( [-(pi/2), 0,  a2,   pi/2   ]);
  L(3)= Link ( [0,     0,    a3,   0      ]);
  L(4)= Link ( [0,     0,    a4,   0      ]);
  L(5)= Link ( [0,     0,    a5,   -(pi/2) ]);
  L(6)= Link ( [0,     0,    a6,   0      ]);

  robot = SerialLink (L);
  robot.name = 'Satellite Servicer';

  deg2rad = pi/180;

% Position 1
disp("************************************")
disp("Case 1: All Joint angles = 0");
q1 = 0 * deg2rad;
q2 = 0 * deg2rad - (pi/2);
q3 = 0 * deg2rad;
q4 = 0 * deg2rad;
q5 = 0 * deg2rad;
q6 = 0 * deg2rad; % radians
robot.plot ([q1, q2, q3, q4, q5, q6])
hold on
pause(5.5)
% disp("Transformation Matrix");
T = robot.fkine([q1, q2, q3, q4, q5, q6]);
inv_T = robot.ikine(T);
```

```
disp("Position #1 from Forward Kinematics");
x_pos = T.t(1,1)
y_pos = T.t(2,1)
z_pos = T.t(3,1)
disp("Joint angles from Inverse Forward Kinematics");
inv_T(2) = inv_T(2)+(pi/2);
disp(inv_T)

% Position 2
disp("************************************")
disp("Case 2: q2 = 90, other angles = 0");
q1 = 0 * deg2rad;
q2 = 90 * deg2rad - (pi/2);
q3 = 0 * deg2rad;
q4 = 0 * deg2rad;
q5 = 0 * deg2rad;
q6 = 0 * deg2rad; % radians
robot.plot ( [ q1, q2, q3, q4, q5, q6 ])
hold on
pause(0.5)
% disp("Transformation Matrix");
T = robot.fkine([ q1, q2, q3, q4, q5, q6 ])
inv_T = robot.ikine(T);
disp("Position #2 from Forward Kinematics");
x_pos = T.t(1,1)
y_pos = T.t(2,1)
z_pos = T.t(3,1)
disp("Joint angles from Inverse Forward Kinematics");
inv_T(2) = inv_T(2)+(pi/2);
disp(inv_T)

% Position 3
disp("************************************")
disp("Case 3: q1, q4 = 90, other angles = 0");
q1 = 90 * deg2rad;
q2 = 0 * deg2rad - (pi/2);
q3 = 0 * deg2rad;
q4 = 90 * deg2rad;
q5 = 0 * deg2rad;
q6 = 0 * deg2rad; % radians
robot.plot ( [ q1, q2, q3, q4, q5, q6 ]);
hold on
pause(0.5)
% disp("Transformation Matrix");
T = robot.fkine([ q1, q2, q3, q4, q5, q6 ])
```

```
inv_T = robot.ikine(T);
disp("Position from Forward Kinematics");
x_pos = T.t(1,1)
y_pos = T.t(2,1)
z_pos = T.t(3,1)
disp("Joint angles from Inverse Forward Kinematics");
inv_T(2) = inv_T(2)+(pi/2);
disp(inv_T)
```

A.4 Robot Workspace plot MATLAB script

```
% era_workspace.m

% ERA D-H parameters
d1 = 0.577;
a2 = -0.224; a3 = -1.572; a4 = -1.572; a5 = -0.224; a6 = -(0.577 +
    0.260);

% Angles from -90 to 90, 10 evenly spaced data points
q1 = linspace(-90,90,10)*pi/180;
q2 = linspace(-90,90,10)*pi/180;
q3 = linspace(-90,90,10)*pi/180;
q4 = linspace(-90,90,10)*pi/180;
q5 = linspace(-90,90,10)*pi/180;
q6 = linspace(-90,90,10)*pi/180;

[Q1,Q2,Q3,Q4,Q5,Q6]=ndgrid(q1,q2,q3,q4,q5,q6); %Grid of angles
    between -90 to 90 degrees

%Using the kinematic equations of the ERA robot arm
x_ee = round(a4.*cos(Q4).*(sin(Q1).*sin(Q3) +
    cos(Q1).*cos(Q3).*sin(Q2)) + a4.*sin(Q4).*(cos(Q3).*sin(Q1) -
    cos(Q1).*sin(Q2).*sin(Q3)) +
    a6.*cos(Q6).*(cos(Q5).*(cos(Q4).*(sin(Q1).*sin(Q3) +
    cos(Q1).*cos(Q3).*sin(Q2)) + sin(Q4).*(cos(Q3).*sin(Q1) -
    cos(Q1).*sin(Q2).*sin(Q3))) +
    sin(Q5).*(cos(Q4).*(cos(Q3).*sin(Q1) -
    cos(Q1).*sin(Q2).*sin(Q3)) - sin(Q4).*(sin(Q1).*sin(Q3) +
    cos(Q1).*cos(Q3).*sin(Q2)))) + a2.*cos(Q1).*sin(Q2) +
    a5.*cos(Q5).*(cos(Q4).*(sin(Q1).*sin(Q3) +
    cos(Q1).*cos(Q3).*sin(Q2)) + sin(Q4).*(cos(Q3).*sin(Q1) -
    cos(Q1).*sin(Q2).*sin(Q3))) + a3.*sin(Q1).*sin(Q3) +
    a5.*sin(Q5).*(cos(Q4).*(cos(Q3).*sin(Q1) -
    cos(Q1).*sin(Q2).*sin(Q3)) - sin(Q4).*(sin(Q1).*sin(Q3) +
    cos(Q1).*cos(Q3).*sin(Q2))) + a3.*cos(Q1).*cos(Q3).*sin(Q2) +
    a6.*cos(Q1).*cos(Q2).*sin(Q6));

y_ee = round(a2.*sin(Q1).*sin(theta2) -
    a4.*sin(Q4).*(cos(Q1).*cos(Q3) + sin(Q1).*sin(theta2).*sin(Q3))
    - a3.*cos(Q1).*sin(Q3) -
    a5.*cos(Q5).*(cos(Q4).*(cos(Q1).*sin(Q3) -
    cos(Q3).*sin(Q1).*sin(theta2)) + sin(Q4).*(cos(Q1).*cos(Q3) +
    sin(Q1).*sin(theta2).*sin(Q3))) -
```

```
    a6.*cos(Q6).*(cos(Q5).*(cos(Q4).*(cos(Q1).*sin(Q3) -
    cos(Q3).*sin(Q1).*sin(theta2)) + sin(Q4).*(cos(Q1).*cos(Q3) +
    sin(Q1).*sin(theta2).*sin(Q3))) +
    sin(Q5).*(cos(Q1).*cos(Q3).*cos(Q4) - cos(Q1).*sin(Q3).*sin(Q4)
    + cos(Q3).*sin(Q1).*sin(theta2).*sin(Q4) +
    cos(Q4).*sin(Q1).*sin(theta2).*sin(Q3))) -
    a4.*cos(Q4).*(cos(Q1).*sin(Q3) - cos(Q3).*sin(Q1).*sin(theta2))
    - a5.*sin(Q5).*(cos(Q1).*cos(Q3).*cos(Q4) -
    cos(Q1).*sin(Q3).*sin(Q4) +
    cos(Q3).*sin(Q1).*sin(theta2).*sin(Q4) +
    cos(Q4).*sin(Q1).*sin(theta2).*sin(Q3)) +
    a3.*cos(Q3).*sin(Q1).*sin(theta2) +
    a6.*cos(theta2).*sin(Q1).*sin(Q6));

z_ee = round(d1 - a2.*cos(theta2) - a3.*cos(theta2).*cos(Q3) +
    a6.*sin(theta2).*sin(Q6) - a4.*cos(theta2).*cos(Q3).*cos(Q4) +
    a4.*cos(theta2).*sin(Q3).*sin(Q4) -
    a5.*cos(theta2).*cos(Q3).*cos(Q4).*cos(Q5) +
    a5.*cos(theta2).*cos(Q3).*sin(Q4).*sin(Q5) +
    a5.*cos(theta2).*cos(Q4).*sin(Q3).*sin(Q5) +
    a5.*cos(theta2).*cos(Q5).*sin(Q3).*sin(Q4) +
    a6.*cos(theta2).*cos(Q3).*cos(Q6).*sin(Q4).*sin(Q5) +
    a6.*cos(theta2).*cos(Q4).*cos(Q6).*sin(Q3).*sin(Q5) +
    a6.*cos(theta2).*cos(Q5).*cos(Q6).*sin(Q3).*sin(Q4) -
    a6.*cos(theta2).*cos(Q3).*cos(Q4).*cos(Q5).*cos(Q6));

plot3(x_ee(:),y_ee(:),z_ee(:),'.')

xlabel('X')
ylabel('Y')
zlabel('Z')
```