University of Mumbai

Analysis and Simulation of Robotic Arms: SCARA & UR5

Submitted in partial fulfillment of requirements
For the degree of

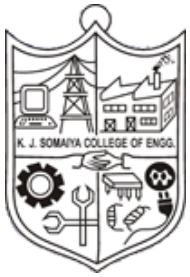
Bachelors in Technology

by

Saeesh Aher
Roll no. 1712001
Harmanjeet Singh Bilkhu
Roll no. 1712005
Dhairya Maisheri
Roll no. 1712027

Guide

Prof. Annu Abraham



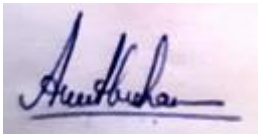
Department of Electronics Engineering
K. J. Somaiya College of Engineering, Mumbai-77
(Autonomous College Affiliated to University of Mumbai)
Batch 2017 -2021

K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

Certificate

This is to certify that the dissertation report entitled **Analysis and Simulation of Robotic Arms: SCARA & UR5** is bonafide record of the dissertation work done by **Saeesh Aher, Harmanjeet Singh Bilkhu and Dhairya Maisheri** in the year 2020-21 under the guidance of Prof. Annu Abraham of Department of Electronics Engineering in partial fulfillment of requirement for the Bachelors in Technology degree in Electronics Engineering of University of Mumbai.



Annu Abraham
Guide

_____
Head of the Department

_____
Principal

Date: 02/06/21

Place: Mumbai-77

K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

Certificate of Approval of Examiners

We certify that this dissertation report entitled **Analysis and Simulation of Robotic Arms: SCARA & UR5** is bona fide record of project work done by **Saeesh Aher, Harmanjeet Singh Bilkhu and Dhairya Maisheri**.

This project is approved for the award of Bachelors in Technology Degree in Electronics Engineering of University of Mumbai.

Annu Abraham

Internal Examiner

Rajdeep Gupta

External Examiner

Date:02/06/21

Place: Mumbai-77

K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We declare that we have properly and accurately acknowledged all sources used in the production of this report. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission.

We understand that any violation of the above will be a cause for disciplinary action by the college and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____ **Saeesh Aher** **1712001**	_____ **Harmanjeet Singh Bilkhu** **1712005**
_____ **Dhairya Maisheri** **1712027**	

Date:02/06/21

Place: Mumbai-77

Dedicated to family, teachers and friends. . .

Abstract

Robotic arms have become an integral part of today's world. They have revolutionized industries and automotive manufacturing, helped scientists in space exploration and have assisted medical surgeries. It became possible due to the advancements that have taken place in recent years. In order to understand and work on modern techniques, one should first understand the fundamental theory behind robotics. The aim of the project was to understand the fundamental theory and then perform the analysis and control of two robotic arms: SCARA and UR5, with the help of simulations. We modelled SCARA in MATLAB Simulink using the Simscape toolbox and completed kinematical equations verification, workspace analysis and different task simulations using point-to-point motion control. We then configured and controlled SCARA and UR5 in ROS using MoveIt! motion planning framework. Through packages of ROS and customized scripts, we simulated pick and place, straight line traversal, object tracking and other operations. Together with technical details, the aim is also to provide readers with tools for the simulation and control of robotic arms.

Keywords: Robotic Arms, SCARA, UR5, MATLAB/Simulink, ROS, MoveIt!

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

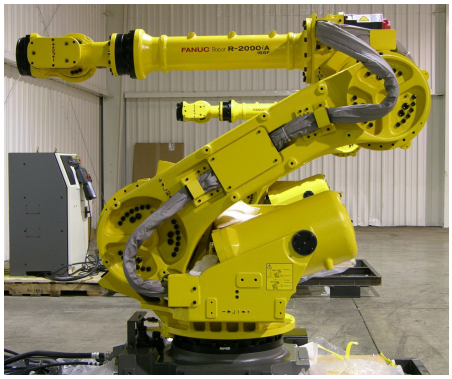
1.1 Background

The field of robotics has seen significant advancements in past years. Today we are exploring other planetary surfaces, ocean depths, outer space with the help of robots. Industrial automation, prosthetics, medical surgeries, 3d printing are few applications where robots have proved very useful. Further robotic arms are essential part of the International Space Station (ISS) and Mars rovers where they perform maintenance as well as experimental tasks.

Modern techniques of computer science like motion planning, generic kinematics, collision avoidance and trajectory processing are used in these robots. To work on them one has to first understand the fundamental theory behind robotic arms which involves direct and inverse kinematics, trajectory planning and control. This project aims to understand the fundamental theory behind robotic arms and then explore the recent advancements by doing simulations. The project is based on two robotic arms: kinematically simple 4-axis SCARA and 6-axis UR5.

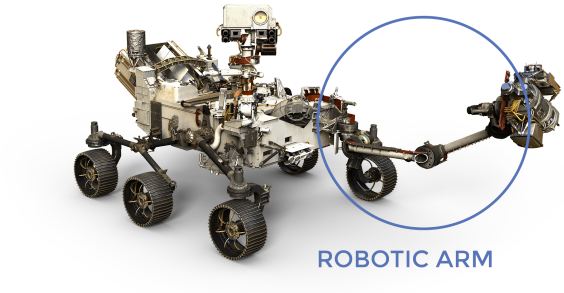
1.2 Objectives

- To first get acquainted with the fundamentals concepts and then complete the analysis and control of both the robotic arms: SCARA and UR5
- To use tools like MATLAB/Simulink, ROS and Gazebo for simulation purposes
- To perform different applications like pick and place, straight line motion and others in simulation



(a) A Industrial Robot

(b) Robotic arm on ISS



(c) Robotic arm on Perseverance Rover

Figure 1.1: Robotic Arms

- To give readers a overview of the theoretical analysis of the two robotic arms and capabilities of the simulation softwares, along with references

1.3 Outline of the report

In this chapter we introduce the topic and objectives of the project. In chapter 2, basic definitions and an overview of SCARA and UR5 arms is given. Chapter 3 deals with the theoretical analysis of the arms which includes direct and inverse kinematics, workspace analysis, trajectory planning and control. In chapter 4, theoretical verification completed and simulations performed in MATLAB/Simulink and ROS is given. Finally, we conclude with chapter 5.

Chapter 2

Literature Survey

Ample amount of literature is available in the form of text and video. Several textbooks like Schilling (1996) and Craig (2009) are available for it. We began with reading the Schilling (1996) textbook and understood it thoroughly in the initial period of the project.

In this chapter we first specify few basic definitions in robotics. The reader can refer the textbooks for detailed explanation. We then discuss about SCARA and UR5 arms followed by an overview of the simulation tools used.

2.1 Basic Definitions

Robotic Arm

A robotic arm is a type of mechanical arm, usually programmable, with similar functions to a human arm; the arm may be the sum total of the mechanism or may be part of a more complex robot. It can be modelled as a chain of rigid links interconnected by joints (Figure 2.1). One end is fixed called the base while the other is free to move called end-effector or simply tool. There are two basic types of joints:

Prismatic (P): Linear motion about an axis

Revolute (R): Rotary motion about an axis

Direct Kinematics

It refers to calculating the position and orientation of the end-effector given the values of the joint variables.

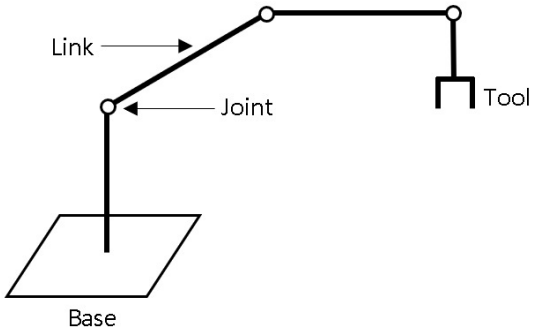


Figure 2.1: Robot Arm

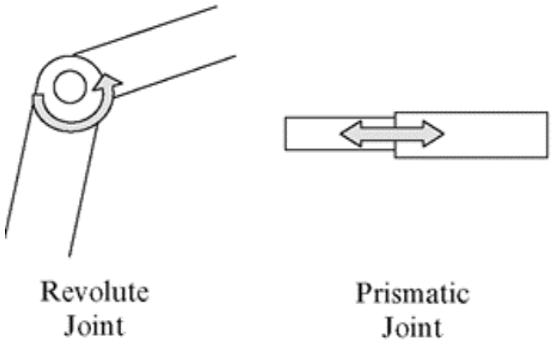


Figure 2.2: Two basic types of joints

Inverse Kinematics

It refers to calculating the joint variables given the desired position and orientation of the end-effector.

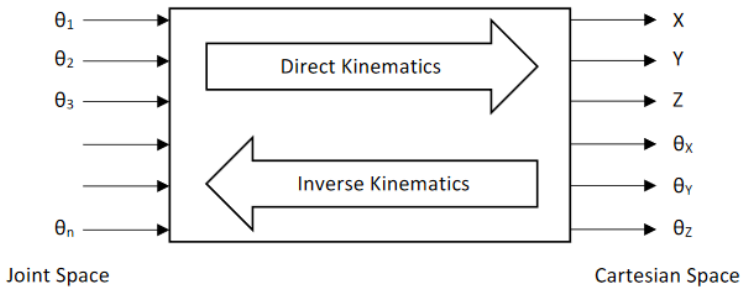


Figure 2.3: Relationship between direct and inverse kinematics

D-H Parameters

Denavit-Hartenberg algorithm [1] is used to assign right handed orthonormal co-ordinate frames to the links of the robot arm. In the link co-ordinate diagram, the four kinematic parameters namely joint angle (θ), joint distance (d), link length (a) and link twist angle (α) can be assigned. Then, homogenous co-ordinate transformation matrix can be used to formulate relationship between the adjacent frames and consequently between the base and the end-effector.

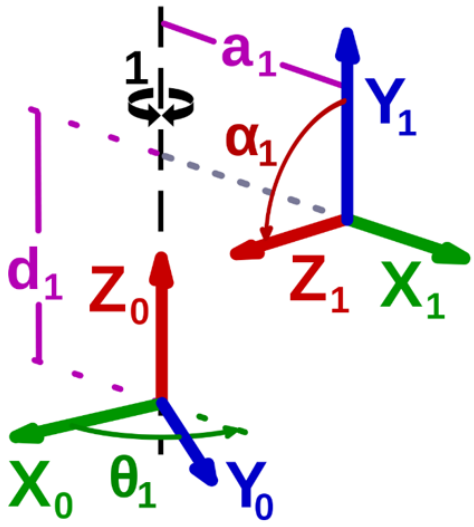


Figure 2.4: Denavit-Hartenberg Transformation

Arm Matrix

Let $\{L_0, L_1, ..., L_n\}$ be a set of link co-ordinates frames assigned using D-H algorithm. Then the transformation of link L_k to $L_k - 1$ is given by

$$T_{k-1}^{k} = \begin{bmatrix} C\theta_k & -C\alpha_k S\theta_k & S\alpha_k S\theta_k & a_k C\theta_k \\ S\theta_k & C\alpha_k C\theta_k & -S\alpha_k C\theta_k & a_k S\theta_k \\ 0 & S\alpha_k & C\alpha_k & d_k \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

Example: T_0^1 will map shoulder co-ordinates into base co-ordinates.
These matrices can be multiplied together and the matrix for the entire arm can be obtained, called the arm matrix. It will be of the form:

[1] https://en.wikipedia.org/wiki/Denavit-Hartenberg_parameters

$$T_{base}^{tool}(q) = \begin{bmatrix} & R(q) & & p(q) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Here 3x3 submatrix $R(q)$ specifies the orientation of the tool while 3x1 submatrix $p(q)$ specifies the position of the tool. The arm matrix is the solution to the direct kinematics problem.

Tool Configuration Vector

The desired tool position and orientation can be specified in a number of ways. Out of which, tool configuration vector is a minimal representation, having only 6 components in general. Let p and R denote the position and orientation of the tool relative to the base frame and q_n represents the tool roll angle. Then the tool configuration vector (w) in R^6 is defined as

$$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} p \\ [exp(q_n/\pi)]r^3 \end{bmatrix} \tag{2.3}$$

Here r^3 is the approach vector of the tool.

Joint Space Work Envelope

The set of all the values that the joint variables can assume is called joint space work envelope.

Work Envelope

The locus of all the points that are reachable by the tool tip is called the work envelope of the robot arm.

2.2 SCARA

Selective Compliance Assembly Robot Arm (SCARA) is four axis robotic arm. It is a general class of robots where all the joint axis are vertical. It was first invented by the Japanese scientist Hiroshi Makino from Yamanashi University (Figure 2.5) (Gasparetto and Scalera, 2019). The first commercially available SCARA robot was the Adept One Robot shown in Figure 2.6. Today companies like Epson, FANUC, Yamaha and many more, manufacture high performance SCARA robots (Figure 2.7).

Figure 2.5: One of the first prototypes of SCARA robot, designed by Hiroshi Makino

Figure 2.6: Examples of AdeptOne SCARA robots

(a) Epson's (b) FANUC's (c) Yamaha's

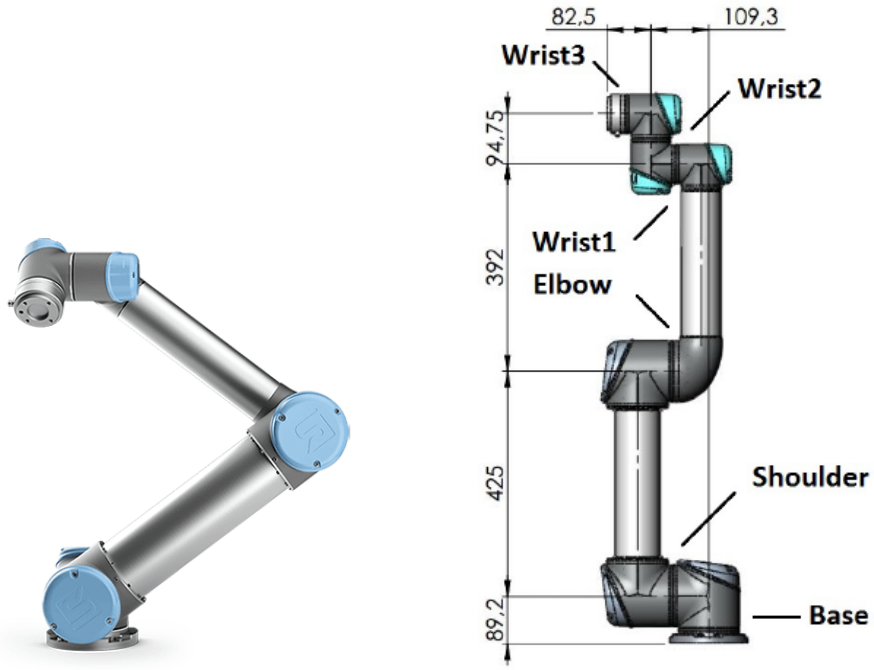
Figure 2.7: SCARA robots

These robots are among the most widely used robots in industry due to simple nature and high accuracy. Selective compliant means it is compliant in the X-Y axis, and rigid in the Z-axis. Therefore, they are majorly used in vertical assembly applications like PCB assembling and drilling. They are also used in applications like pick and place and stacking.

This robot is kinematically simple to analyze and thus it helps a novice to form a good understanding of the subject. The joint configuration is **R-R-P-R**. The first two joints determine the radial position of the tool tip, third joint determines the height and fourth joint determines the tool roll orientation.

2.3 UR5

UR5 is a lightweight six axis industrial robot shown in Figure 2.8. It is developed by the company Universal Robots[2]. Six-axis robots have more directional control and larger work envelope than SCARAs, making it suitable for complex tasks like grabbing something under a table. It can execute operations like assembly, material handling, welding and others.



(a) UR5 Robot (b) Structure of UR5

Figure 2.8: UR5

[2]https://www.universal-robots.com/products/ur5-robot/

Table 2.1: Specifications of UR5

Weight	18.4 kg / 40.6 lbs
Payload	5 kg / 11 lbs
Joint ranges	+/- 360°
Degrees of freedom	6 rotating joints
Reach	850 mm / 33.5 in
Footprint	Ø149 mm / 5.9 in

The six joints are referred to as Base, Shoulder, Elbow, Wrist1, Wrist2 and Wrist3. The shoulder and elbow joints are rotating perpendicular to the Base joint. The first three joints are connected with long links to have a long reach. The remaining three wrist revolute joints are used to orient the tool. The kinematical analysis is complex and requires careful motion planning to avoid collisions with itself and the environment.

2.4 Simulation tools

Following tools/ softwares are used for verification of equations, visualization and motion planning of robotic arm. We only present the overview, the reader can refer to the footnote links for better understanding and tutorials.

2.4.1 MATLAB/ Simulink

Simulink, an add-on product to MATLAB, provides an interactive, graphical environment for modeling, simulating, and analyzing of dynamic systems.[3] For modeling, Simulink provides a graphical user interface (GUI) for building models as block diagrams. In it, libraries such as Simscape Multibody[4] (formerly SimMechanics) provides a multibody simulation environment for 3D mechanical systems, such as robots, vehicle

[3] https://www.mathworks.com/products/simulink.html
[4] https://www.mathworks.com/products/simscape-multibody.html

suspensions, construction equipment, and aircraft landing gear. We can import complete CAD assemblies, and an automatically generated 3D animation lets us visualize the system dynamics.

2.4.2 ROS

Robot Operating System (ROS) is an open source collection of software frameworks for robot software development. The aim is to provide a standard for robotics software development, that we can use on any robot. ROS is more of a middleware which handles the communication between programs in a distributed system. We can create sub programs one for each task and run it without the whole application.

C++ and Python are the most preferred languages when developing robotics applications. A user can use existing ROS libraries directly in the code. Several open source packages are available for motion planing, collision avoidance, mapping, 3d visualization and more. One such library is MoveIt![5] which is used for manipulation and has been used on over 150 robots. Being an open source community, anyone can publish and share his/her work or improvements. Reader can refer these links[6,7,8] for in-depth tutorials and further exploration.

Gazebo

Gazebo[9] is an open-source, free 3D robotics simulator. It can be used to simulate robots in complex indoor and outdoor environments. ROS can be integrated with Gazebo to test our code on robot.

[5] https://moveit.ros.org/
[6] http://wiki.ros.org/
[7] https://answers.ros.org/questions/
[8] https://github.com/ros
[9] http://gazebosim.org/

Chapter 3

Theoretical Analysis

In this chapter we will establish the mathematical foundation for analysing robotic arms. We will start with the direct and inverse kinematics. Once kinematical solutions are set up, we will analyse the workspace and plan the tool tip trajectory in it.

3.1 Direct and Inverse Kinematics

Direct Kinematics refers to determining the position and orientation of the tool given the vector of joint variables. This problem can be solved by using the Denavit-Hartenberg (D-H) algorithm which leads to the arm matrix. Arm matrix represents the position p and orientation R of the tool in the base frame as a function of the joint variables q. Inverse kinematics refers to determining the joint variables given a desired position and orientation of the tool. It can be solved analytically or numerically.

3.1.1 SCARA

Using D-H algorithm, the link co-ordinate diagram of SCARA is shown in Figure 3.1 and D-H parameters are given in Table 3.1. Both of these indicate that SCARA is a kinematically simple robot and easier to analyze.

Table 3.1: D-H parameters of four axis SCARA

Axis	θ	d	a	α
1	q_1	d_1	a_1	π
2	q_2	0	a_2	0
3	0	q_3	0	0
4	q_4	d_4	0	0

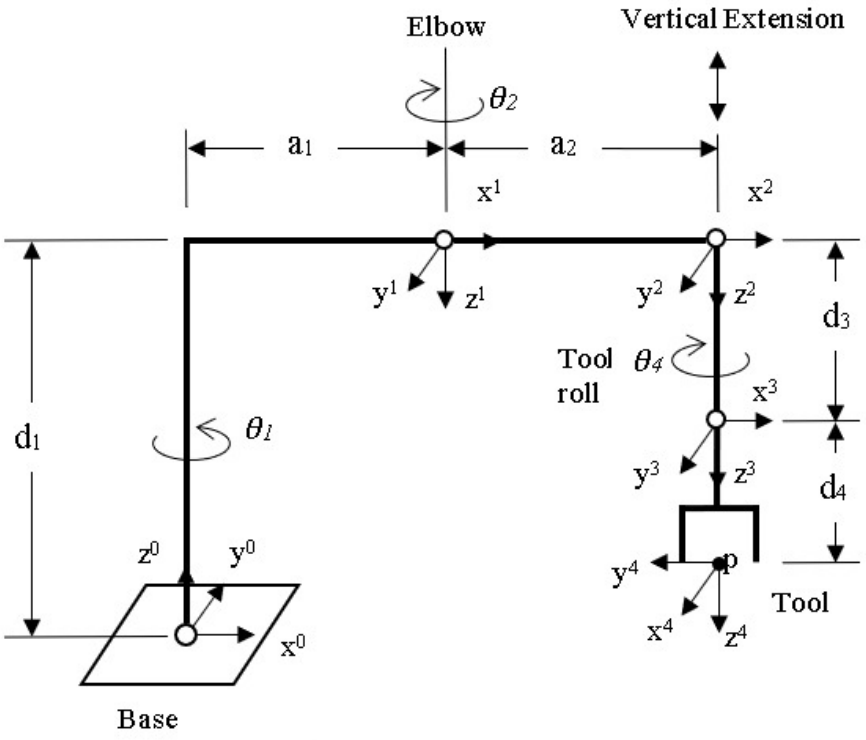


Figure 3.1: Link co-ordinates of four axis SCARA

Using the general arm matrix, the arm matrix for SCARA can be constructed as follows:

$$
\begin{aligned}
T_{base}^{tool} &= T_0^1 T_1^2 T_2^3 T_3^4 \\
&= \begin{bmatrix} C_1 & S_1 & 0 & a_1C_1 \\ S_1 & -C_1 & 0 & a_1S_1 \\ 0 & 0 & -1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_2 & -S_2 & 0 & a_2C_2 \\ S_2 & C_2 & 0 & a_2S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_4 & -S_4 & 0 & 0 \\ S_4 & C_4 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
T_{base}^{tool} &= \begin{bmatrix} C_{1-2-4} & S_{1-2-4} & 0 & a_1C_1 + a_2C_{1-2} \\ S_{1-2-4} & -C_{1-2-4} & 0 & a_1S_1 + a_2S_{1-2} \\ 0 & 0 & -1 & d_1 - q_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{3.1}
$$

where $C_k = cos\theta_k$, $C_{kj} = cos(\theta_k + \theta_j)$, $C_{k-j} = cos(\theta_k - \theta_j)$ and similar.

It is to be noticed that the approach vector is fixed at $r_3 = -i^3$. this is the characteristic of SCARA robots, which are designed to manipulate objects from directly above.

Now lets discuss the analytical solution for its inverse kinematics. From Equation (3.1), we can write the tool configuration vector for SCARA as

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \end{bmatrix} = \begin{bmatrix} a_1C_1 + a_2C_{1-2} \\ a_1S_1 + a_2S_{1-2} \\ d_1 - q_3 - d_4 \\ 0 \\ 0 \\ -\exp(q_4/\pi) \end{bmatrix} \tag{3.2}$$

We notice that the radial position of the tool tip is dependent only on q_1 and q_2 joint variables. This pair of simultaneous equations can be solved to get q_1 and q_2. q_3 and q_4 can be solved directly. Thus the inverse kinematics solution of SCARA robot can be given as:

$$\begin{aligned} q_2 &= \pm arccos\left(\frac{w_1^2 + w_2^2 - a_1^2 - a_2^2}{2a_1a_2}\right) \\ q_1 &= atan2\left[a_2S_2w_1 + (a_1 + a_2C_2)w_2, (a_1 + a_2C_2)w_1 - a_2S_2w_2\right] \\ q_3 &= d_1 - d_4 - w_3 \\ q_4 &= \pi\ ln\ |w_6| \end{aligned} \tag{3.3}$$

3.1.2 UR5

Using D-H algorithm, the link co-ordinate diagram of UR5 arm is shown in Figure 3.2 and D-H parameters are shown in Table 3.2. The arm matrix for UR5 can be written as:

$$T_{base}^{tool} = T_0^1T_1^2T_2^3T_3^4T_4^5T_5^6 \tag{3.4}$$

Table 3.2: D-H paramters of six axis UR5

Axis	θ	d	a	α
1	q_1	d_1	0	$\pi/2$
2	q_2	0	a_2	0
3	q_3	0	a_3	0
4	q_4	d_4	0	$\pi/2$
5	q_5	d_5	0	$-\pi/2$
6	q_6	d_6	0	0

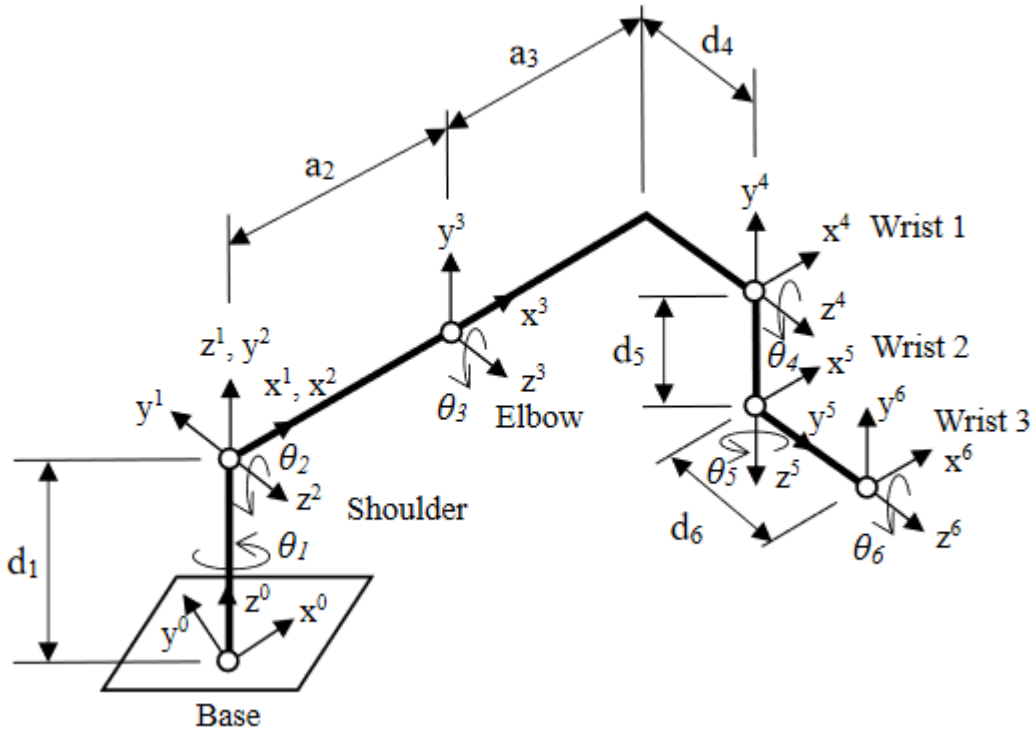


Figure 3.2: Link co-ordinates of six axis UR5

The general arm matrix and D-H parameters can be used to write the transformations for each link. Then the complete tranformation can be derived using Equation (3.4). The complete analytic equations can be found in Hawkins (2013).

As noted before, inverse kinematics can be solved analytically or numerically. The complete analytical solution can be found in Andersen (2018) which explains each step thoroughly. We shall discuss the numerical approach. There are multiple libraries available, in this project we make use of the Kinematics and Dynamics Library (KDL)[1]. It solves for the joint angles by solving the linear least squares problem. This is done over a number of iterations, using Newton-Raphson (NR) for gradient descent minimization. The pseudo-inverse of the Jacobian is determined using Singular Value Decomposition (SVD).

Least Square Problem

Consider an n-jointed kinematic chain. Let $y_1, y_2, ..., y_n$ be the Cartesian coordinates and $x_1, x_2, ..., x_n$ be the joint angles respectively. Each Cartesian coordinate is a function of each of the joint angles,

$$y_1 = f_1(x_1, x_2, ..., x_n)$$

[1] `https://www.orocos.org/kdl.html`

$$y_2 = f_1(x_1, x_2, ..., x_n)$$
$$\vdots$$
$$y_n = f_1(x_1, x_2, ..., x_n)$$

If we take the partial derivatives of each Cartesian coordinate with respect to each joint angle, we get,

$$\begin{aligned}
\delta y_1 &= \tfrac{\delta f_1}{\delta x_1} + \tfrac{\delta f_1}{\delta x_2} + .. + \tfrac{\delta f_1}{\delta x_n} \\
\delta y_2 &= \tfrac{\delta f_2}{\delta x_1} + \tfrac{\delta f_2}{\delta x_2} + .. + \tfrac{\delta f_2}{\delta x_n} \\
&\vdots \\
\delta y_n &= \tfrac{\delta f_n}{\delta x_1} + \tfrac{\delta f_n}{\delta x_2} + .. + \tfrac{\delta f_n}{\delta x_n}
\end{aligned}$$

It can be denoted in matrix form as,

$$\dot{Y} = J(X)\dot{X}, \textit{ where } J(X) \textit{ is the Jacobian of } F \tag{3.5}$$

The solution of (3.5) is that of a linear least squares problem that minimizes $\|\dot{Y} - J(X)\dot{X}\|$

Newton-Raphson Method

The Newton-Raphson Method is one of the methods used for minimization of $\|\dot{Y} - J(X)\dot{X}\|$. It uses the idea that a continuous and differentiable function can be approximated by a straight line tangent to it. In this method for any x-value x_n, the next value is given by

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{3.6}$$

In this method the intersection of the gradient with the primary dimension, which in our case is the Jacobian, is measured. This intersection becomes the guess for the next iteration. This process is repeated until values within tolerance bounds are found.

Singular Value Decomposition

The SVD of an mxn matrix A with real values is given as

$$A = U\Sigma V^T$$

where,

U is an mxm orthogonal matrix

V is an nxn orthogonal matrix

Σ is a diagonal matrix with non-negative real entries on the diagonal containing all of the standardized values for Av divided by their singular values[2]

The pseudo inverse $A^{\dagger}$ can then be written as,

$$A^{\dagger} = V\Sigma^{-1}U^{T}$$

Equation(3.5) can also be written as

$$\dot{\theta} = J(\theta)^{-1}v \tag{3.7}$$

where,

θ are joint angles

$\dot{\theta}$ is nx1 joint velocity vector

v is 6x1 cartesian velocity vector containing a 3x1 linear velocity vector and a 3x1 rotational velocity vector stacked together

Thus for joint angle calculation at an instant, inverse of Jacobian is needed but direct inverse is only possible when no. of joints is equal to D.O.F. in Cartesian Space i.e $J(X)$ is a square matrix. So in case of redundant arms, a pseudo-inverse is used for approximate value of inverse. This is where algorithms like SVD are used.

While setting up the UR5 robot arm using MoveIt Setup Assistant in 4.4.1, we just need to select the KDL as kinematics solver. Thus, it is relatively easy to configure and widely applicable.

3.2 Workspace Analysis

Work envelope is the locus of all the points that are reachable by the tool tip. Knowing the work envelope of a robot arm, we can generate tool trajectory to perform required tasks.

3.2.1 SCARA

From Equation (3.2) we note that the radial position of tool tip is dependent on joint variables q_1 and q_2 only. q_3 will just determine the height of the tool tip while q_4 will

[2] https://medium.datadriveninvestor.com/a%2Dbeginners%2Dguide%2Dto%2Dsingular%2Dvalue%2Ddecomposition%2Dsvd%2Ddba72f782127

provide the tool roll. Let us assume, the joint variables have following constraints:

$$\begin{bmatrix} -\pi \\ -\pi+\beta \\ h \\ -\pi \end{bmatrix} \leq \begin{bmatrix} q_1 & 0 & 0 & 0 \\ 0 & q_2 & 0 & 0 \\ 0 & 0 & q_3 & 0 \\ 0 & 0 & 0 & q_4 \end{bmatrix} \leq \begin{bmatrix} \pi \\ \pi-\beta \\ H \\ \pi \end{bmatrix} \tag{3.8}$$

Here $\beta \geq 0$ will prevent the upper arm and forearm from colliding. Using (3.8), we can plot the the work envelope of the SCARA robot as shown in section 4.2.

3.2.2 UR5

From the specifications, all the six revolute joints have joint ranges as ±360°. Thus the work envelope will be of spherical shape. It is important to note that in the whole sphere it will not be possible for the tool tip to reach certain locations. It is the case because in such situations the robot arm may collide with itself or may go into singularity or given tool orientation is not realisable. Therefore to plot the exact work envelope it is necessary to know the task to be performed. In section 4.2 general work envelope is shown.

3.3 Trajectory Planning and Control

Once the work envelope is known, the next step is to plan the trajectory for the tool tip in cartesian space to perform required tasks. Then command the robot joints to follow that tool tip trajectory. There are two main types to control:

Point to point:

Tool moves through a sequence of discrete knot points in workspace. Path between the points is not user given. Used in pick and place type of applications. We used this method in this project.

Continuous path:

Tool moves along a prescribed path at a given velocity specified by the user. Used in painting, welding type of applications.

3.3.1 Pick and place operation

Four discrete points can be used to perform this operation as shown in Figure 3.3. Using inverse kinematics the joints variables can be found at these points. Then the robot can be commanded to follow them. At pick and place points, tool gripper(claw type or vacuum type) can be actuated to pick and drop the object.

Figure 3.3: Pick and place trajectory

3.3.2 Straight line motion

The aim is to make the tool tip follow a straight line in space. Let w^0 and w^1 denote the initial and final point of straight line respectively. Let the movement be carried out in T seconds, then a general straight line trajectory can be represented as:

$$w(t) = [1 - s(t)]w^0 + s(t)w^1 \qquad 0 \leq t \leq T \tag{3.9}$$

Here $s(t)$ is a differentiable speed distribution function mapping [0,T] into [0,1] with $s(0) = 0$ and $s(T) = 1$. More about this function in section 4.3.2. To follow this trajectory accurately, continuous path motion control (Schilling, 1996) must be used. But for many robots, only point to point control is available. The alternative is to perform approximate straight line motion by discretizing the trajectory into number of knot points. It is desirable to minimize the number of points in a optimal manner. The bounded deviations method proposed by Taylor (1979), also given in Schilling (1996) can be used for selecting knot points.

Chapter 4

Simulation and Results

This chapter contains the description and results of the simulations performed using MATLAB/Simulink and ROS. Simulation video links are given in Appendix A. Moreover, footnote links are given where readers can learn more about the steps involved with the help of tutorials.

4.1 Kinematics Verification

The direct and indirect kinematics equations explained in section 3.1 can be verified by using any mathematical software like MATLAB or Excel. Here we have used Simscape Multibody [1] for SCARA robot and excel sheet[2] for UR5 arm.

4.1.1 SCARA

Modelling

To design the physical structure, 'Solidworks'[3] a CAD software was used. This model is then imported to MATLAB Simulink where we performed simulations and tested our robot. We didn't specifically focused on the structure of the robot as we were only testing it in software and not manufacturing it.

First Iteration

For the first iteration, simple cylinders were used as links and Adept One Robot's dimensions given in Schilling (1996) were followed. The 3d model was then imported

[1] https://www.mathworks.com/products/simmechanics.html

[2] https://www.universal-robots.com/articles/ur/application-installation/dh-parameters-for-calculations-of-kinematics-and-dynamics/

[3] https://www.solidworks.com/

to the Simulink model. Further modifications were done. The model and the robot is shown in Figure 4.1 and Figure 4.2.

Four joints block for SCARA are shown with orange color starting with the base joint. Though modelled, the tool roll joint was not used in this model. The inputs to the model were the x,y and z tool co-ordinates required. The inputs were given to a inverse kinematics block which would then calculate the required amount of joints variables for each joint. These joint variables are fed into the joints block shown with orange boxes. They get actuated correspondingly and the robot arm moves to that point in space. In order to check whether the robot has actually reached the required point in space or not, we have used a Sensor block in Simulink which gives us the relative positions between two frames. Marked by green, are tool co-ordinates which Sensor block has measured. While running the simulation, we were able to get the same values as the input given. Thus, the whole model including the joints, inverse kinematics and certain conventional modifications was verified and worked correctly. We also had implemented a direct kinematics block. It was just for the verification of the direct kinematics equations for the robot. Given the joint variables, it would calculate the tool position. It gives the same result as the Sensor block as expected, only difference being in this case output was calculated by us while in Sensor block Simulink calculated by itself.

Second Iteration

In the second iteration, the CAD model (Figure 4.3) was refined a bit and the same was imported to the Simulink shown in Figure 4.4 and Figure 4.5. Similar procedure as mentioned above was followed to build the model. In this model, the tool was designed as a conventional claw type gripper. Thus, it would enable us to do pick and place operations. All further analysis has been done on this model.

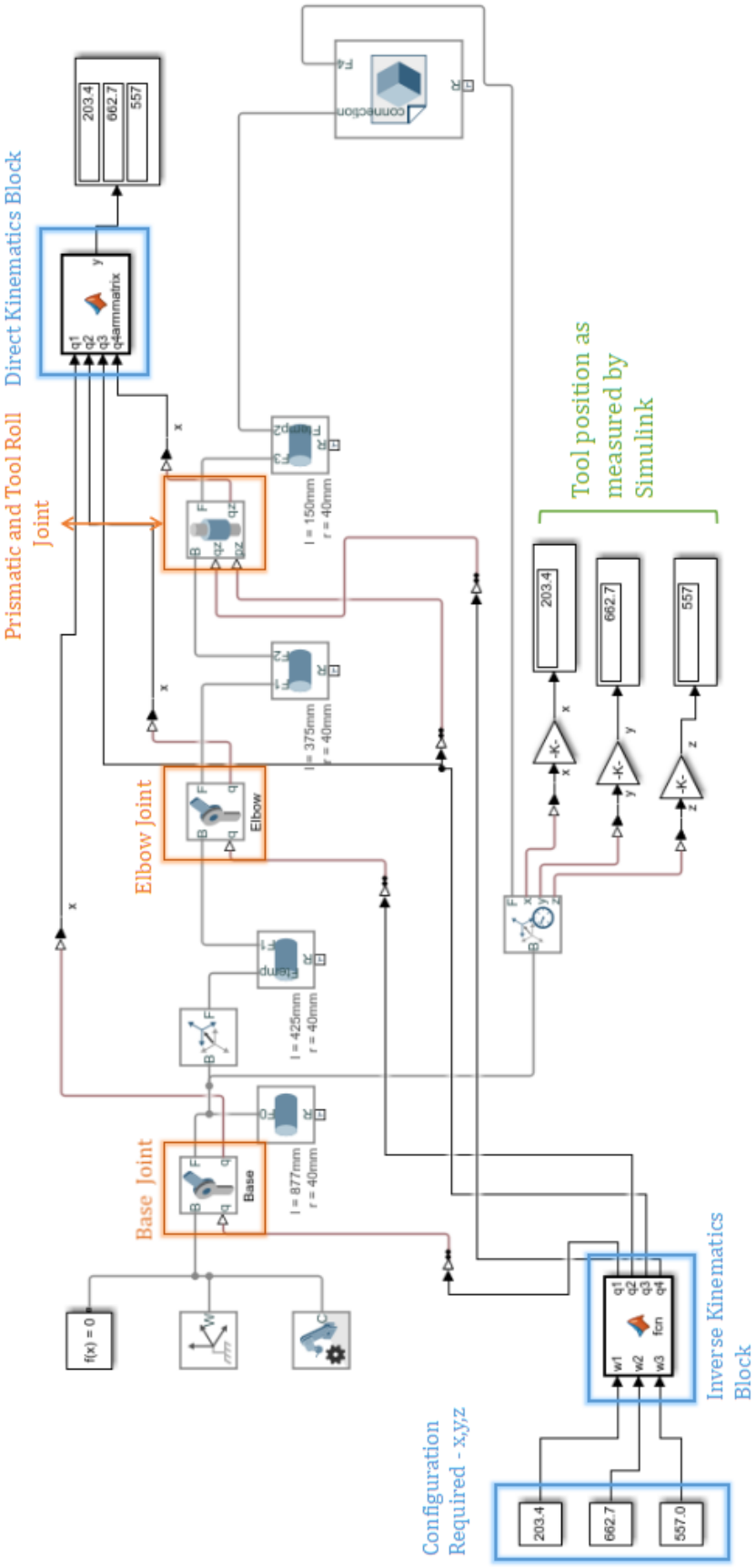
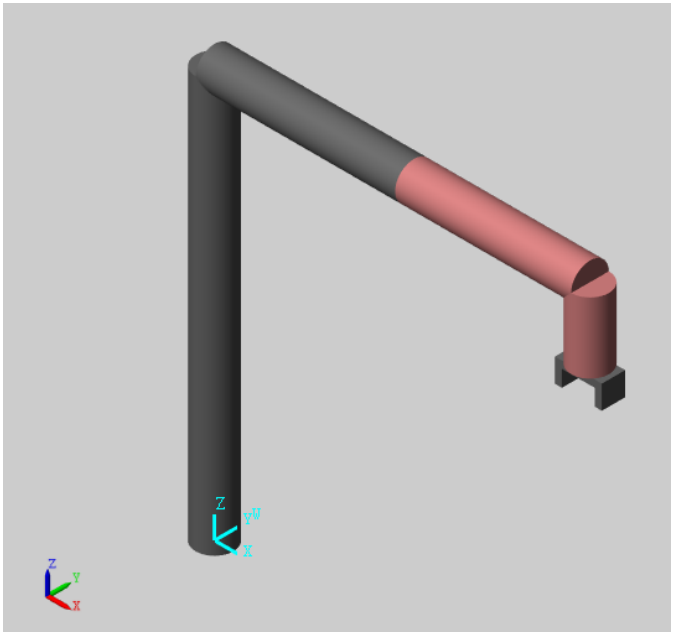
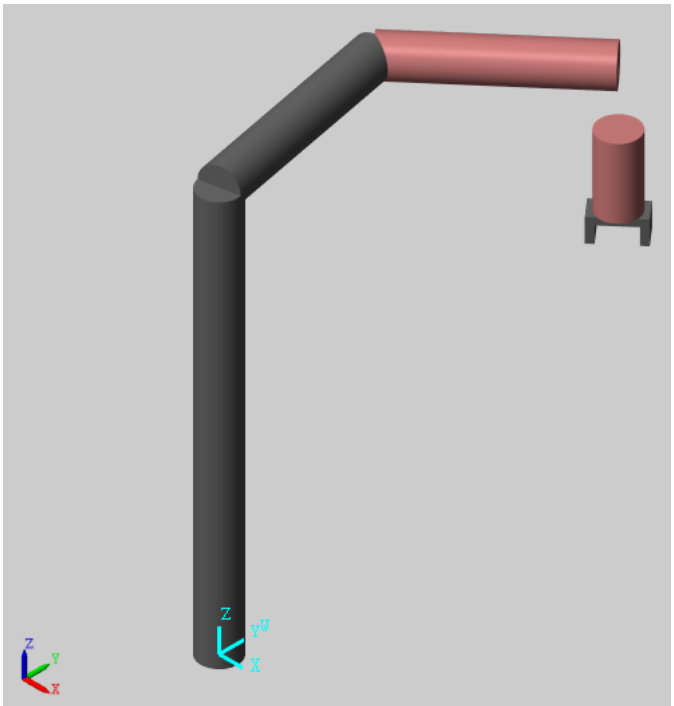


Figure 4.1: Simulink model



(a) Robot Arm in Simulink at initial position



(b) Robot Arm in Simulink at (203.4,662.7,557)mm position

Figure 4.2: Simulink model

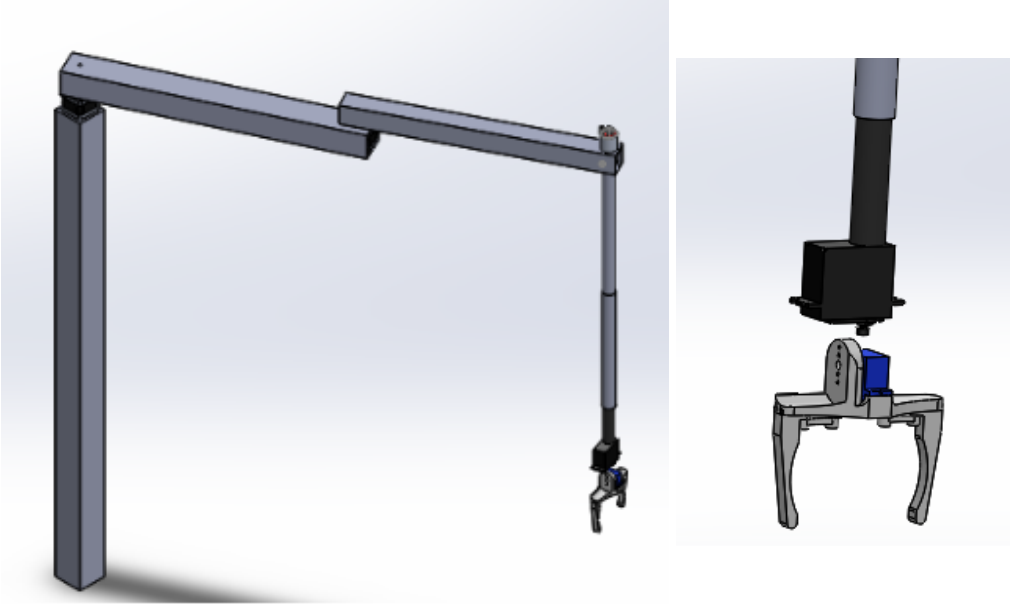


Figure 4.3: CAD model

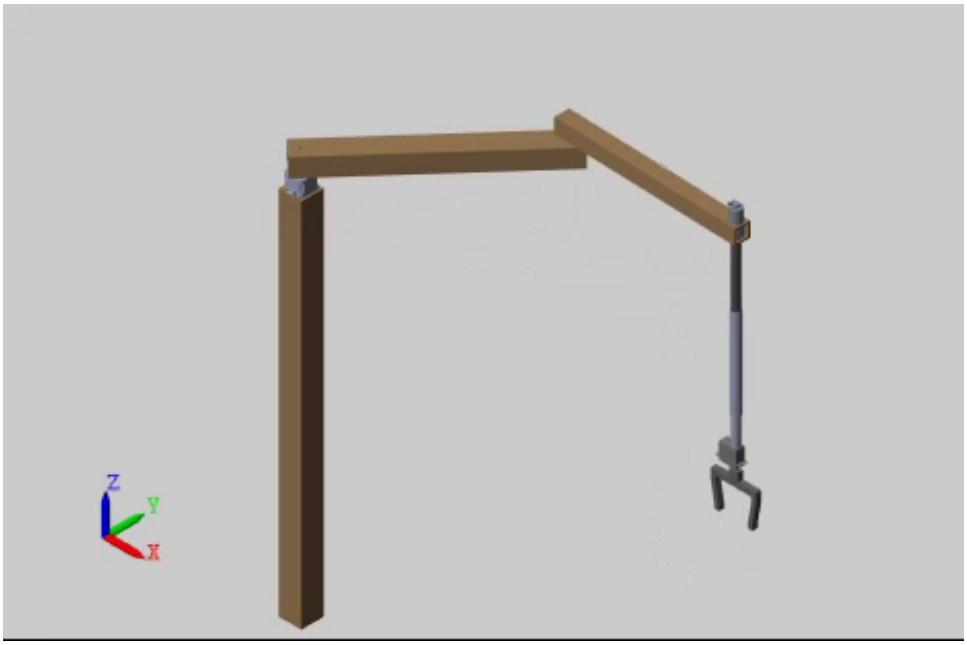


Figure 4.4: Robot Arm in Simulink

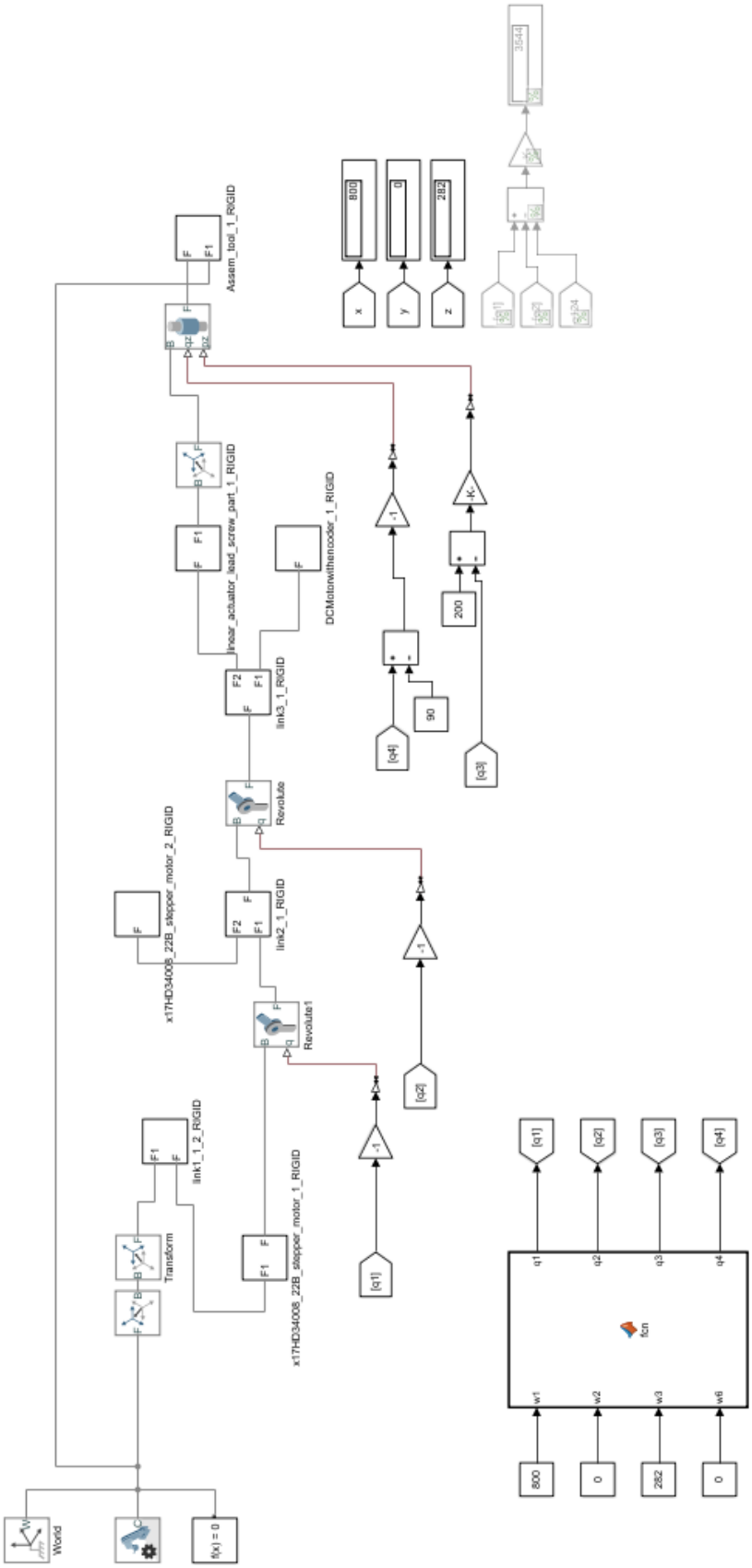


Figure 4.5: Simulink Model

4.1.2 UR5

The visualization of the UR5 forward kinematics can be done using the excel sheet [4] provided by the manufacturer. In the excel sheet, the scroll bars controls the joint angles θ for the respective joints. Then based on the DH parameters tables the intermediate transform matrices are calculated using which the final direct kinematics solution is calculated and visualized using the diagrams.

Since numerical approach was used for inverse kinematics, there was no verification done of the analytical solution.

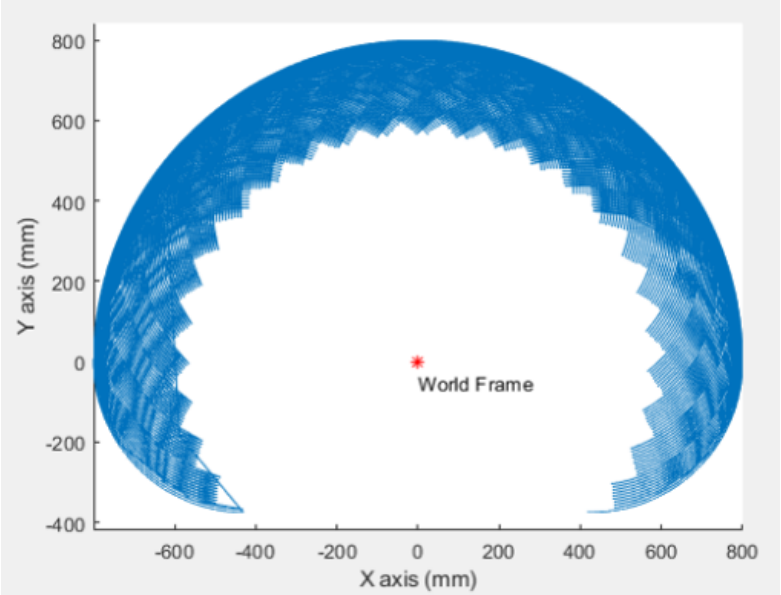
4.2 Workspace

In this section we plot and analyse the robot arm's workspace.

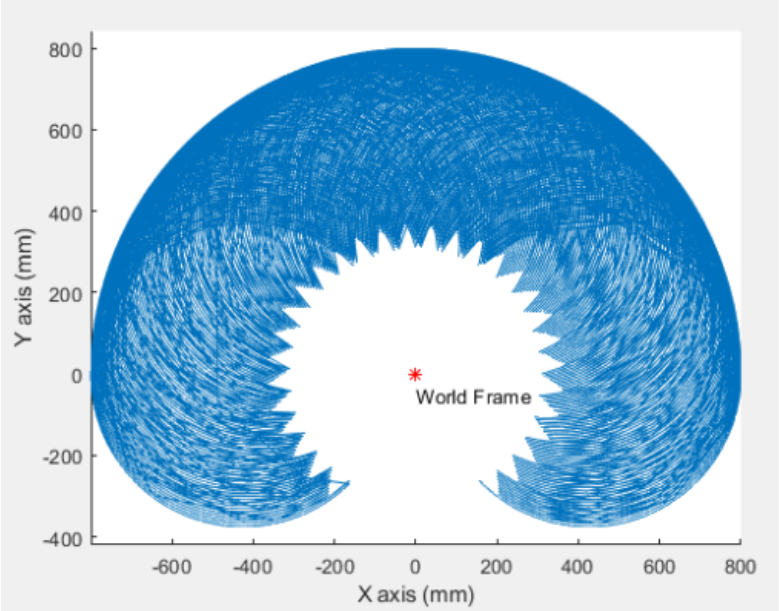
4.2.1 SCARA

The solution for work envelope is robot dependent. In SCARA, ranges for the four joints need to be set in order to develop the work envelope. The horizontal/radial reach of the robot would be limited by the ranges of joints 1 and 2. The vertical reach would be limited by the range of joint 3. And the tool orientation would be limited by joint 4. The focus was on joints 1 and 2 as they determine the space where we could perform the desired applications. We had set the range for joint 1 as $-\pi/2$ *to* $\pi/2$. For joint 2, two ranges were used to plot the work envelope which were $-\pi/2$ *to* $\pi/2$ and $-3\pi/4$ *to* $3\pi/4$. Figure 4.6 show the horizontal cross section of the work envelope. Blue points are points which are within the reach of the robot. From the two plots, it is found that more the range of joint 2, larger will be the work envelope as expected. Hence, we should always look for more freedom in angle ranges while selecting the motor and designing the physical structure. Further, the work envelope can be used while performing any application. For example, in pick place operation, the robot controller can verify whether the object to be picked or the destination location are within the robot's reach. And thus we can use it to evaluate different combinations and choose what fits best for our application.

[4] https://www.universal-robots.com/articles/ur/application-installation/dh-parameters-for-calculations-of-kinematics-and-dynamics/



(a) $q_1 : -\pi/2$ *to* $\pi/2 \quad q_2 : -\pi/2$ *to* $\pi/2$



(b) $q_1 : -\pi/2$ *to* $\pi/2 \quad q_2 : -3\pi/4$ *to* $3\pi/4$

Figure 4.6: SCARA- Horizontal cross section of work envelope

4.2.2 UR5

Since all the six joints are revolute and have ±360°, the work space of UR5 is mostly spherical as shown in Figure 4.7. Within the recommended reach sphere (represented in blue), the robot can move the tool to any position with almost any orientation. When working in the area outside of the recommended reach, but still within the max working area (represented in grey), most positions can be reached but with restrictions on the tool orientation, because the robot physically cannot reach far enough in some situations.

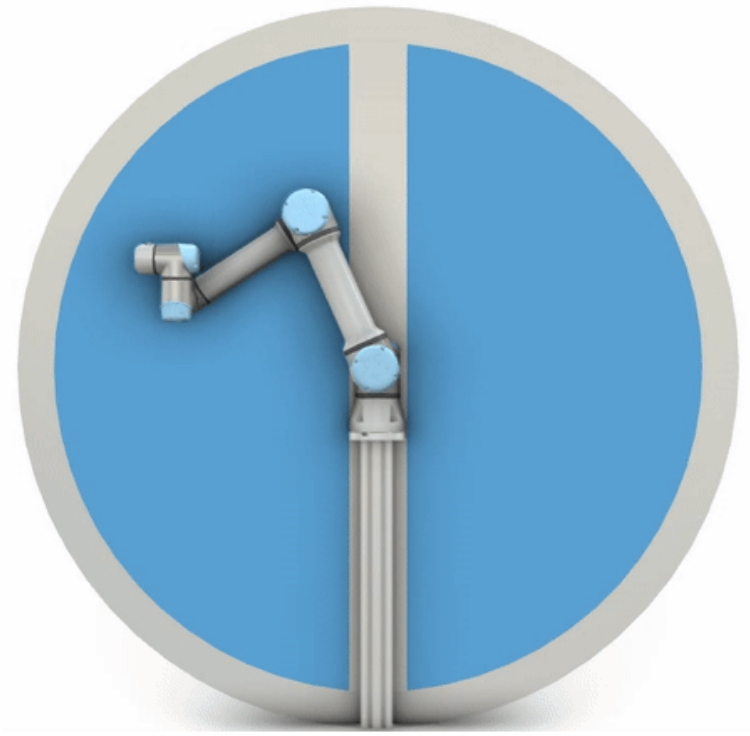


Figure 4.7: UR5 workspace[5]

4.3 SCARA Simulations

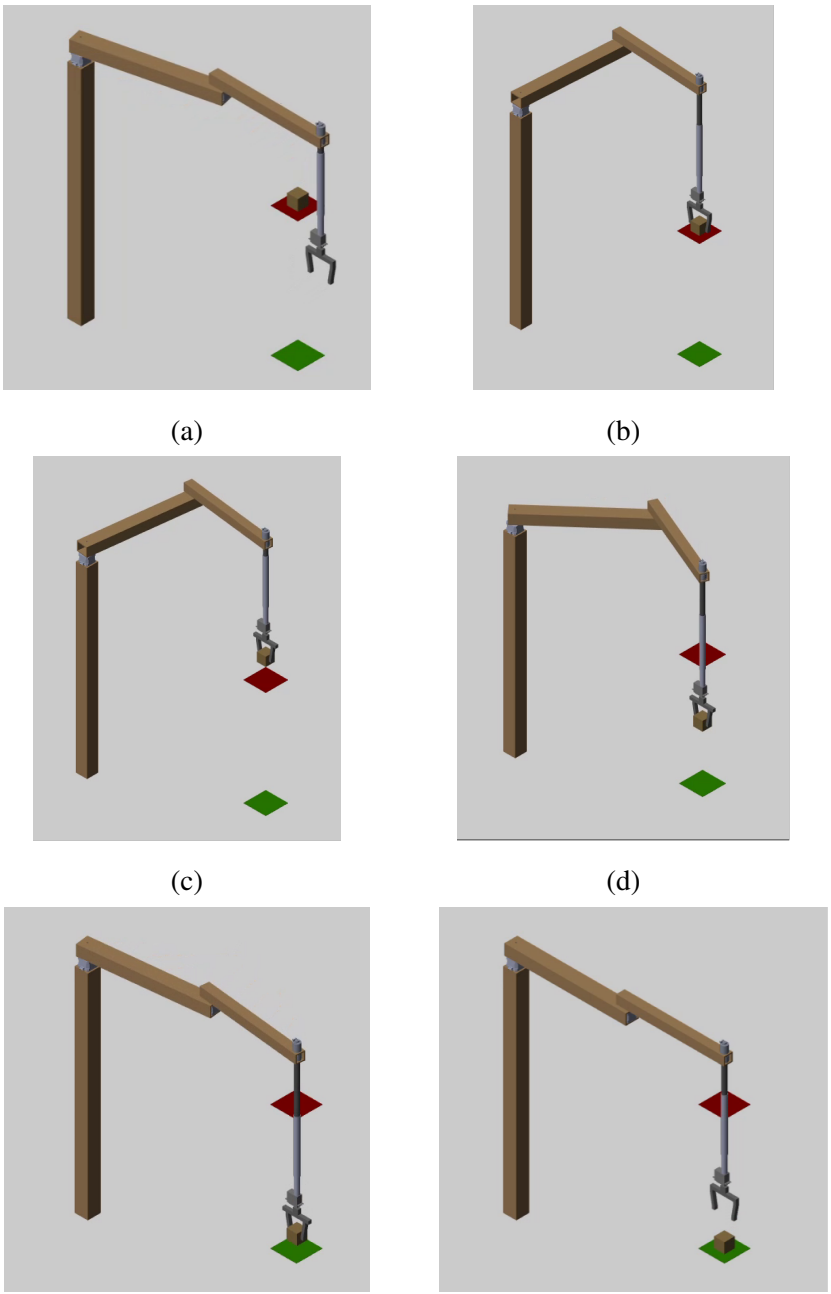
Using the model developed in 4.1.1, we performed simulations which are listed below.

4.3.1 Pick and place operation

The first application that we intended to perform was the pick and place operation. It is the most fundamental task in robotic arms. The aim was to pick the object from w^0 point and place it at w^1 point in tool configuration space. Operation was performed using four

[5]https://www.universal-robots.com/articles/ur/application-installation/what-is-a-singularity/

discrete points as discussed in Figure 3.3. It was performed successfully using the model created in subsection 4.1.1. The snapshots of the simulation are attached in Figure 4.8.



(a) (b) (c) (d) (e) (f)

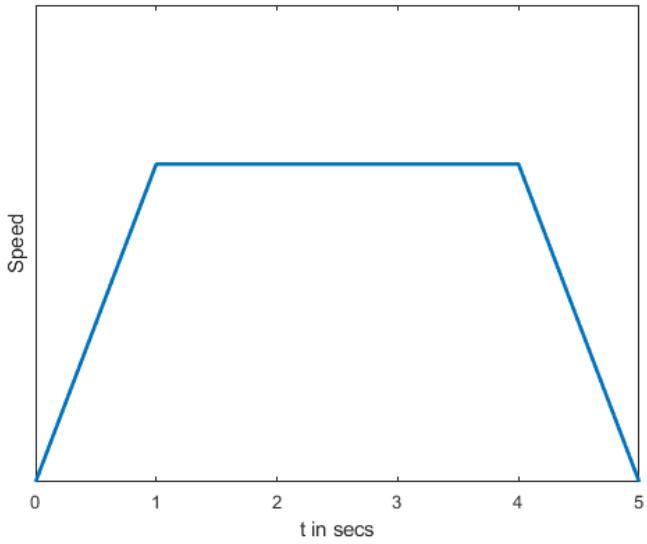
Figure 4.8: SCARA- Pick and place simulation

4.3.2 Straight Line

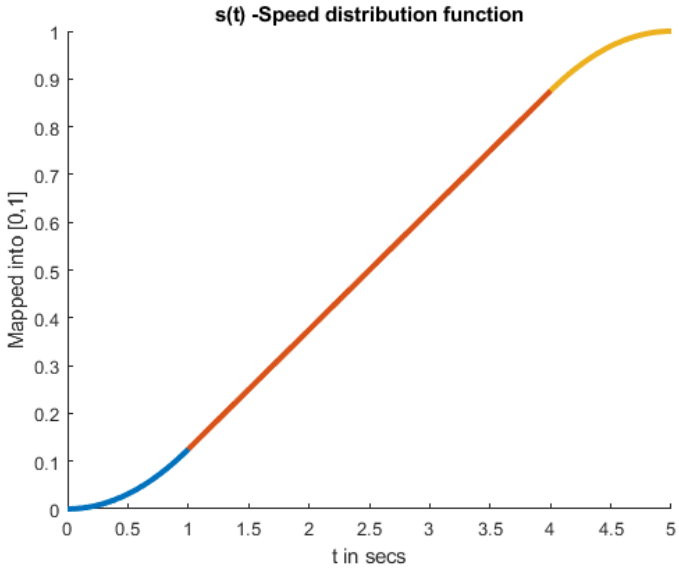
We saw how a straight line trajectory can be planned in subsection 3.3.2.

$$w(t) = [1 - s(t)]w^0 + s(t)w^1 \qquad 0 \leq t \leq T \qquad \text{(3.9 revisited)}$$

The speed distribution function in Equation (3.9) is responsible for the tool tip velocity ($\dot{s}(t)$) along the straight line. Let $T = 5s$. The aim was to ramp the tool velocity to a maximum velocity, then proceed at this speed and finally ramp down to zero velocity as shown in Figure 4.9(a). The corresponding $s(t)$ is shown in Figure 4.9(b). Thus,



(a) Desired tool tip speed profile



(b) Corresponding speed distribution function

Figure 4.9: Speed profile and distribution function

Equation (3.9) was ready to be used. If continuous path control is available, it can be used directly. Else the whole trajectory can be divided into a number of knot points as discussed in subsection 3.3.2.

Using this approach, a sample straight line traversed by the tool tip in catesian space is shown in Figure 4.10. A rectangle box traversed by the tool tip using four straight lines is shown in Figure 4.11. Thus, using combination of straight lines, multiple shapes can be obtained.

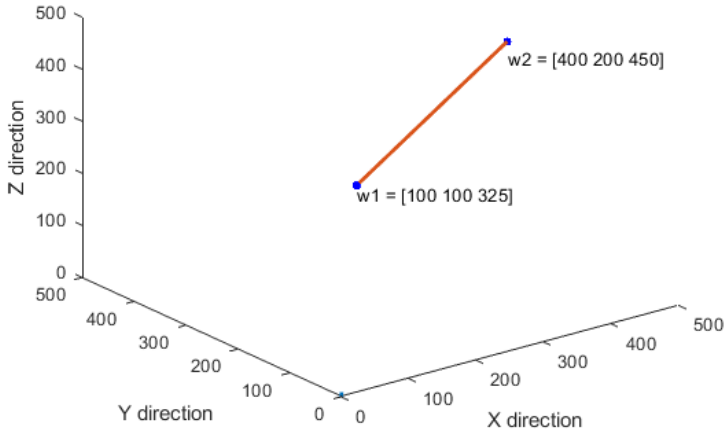
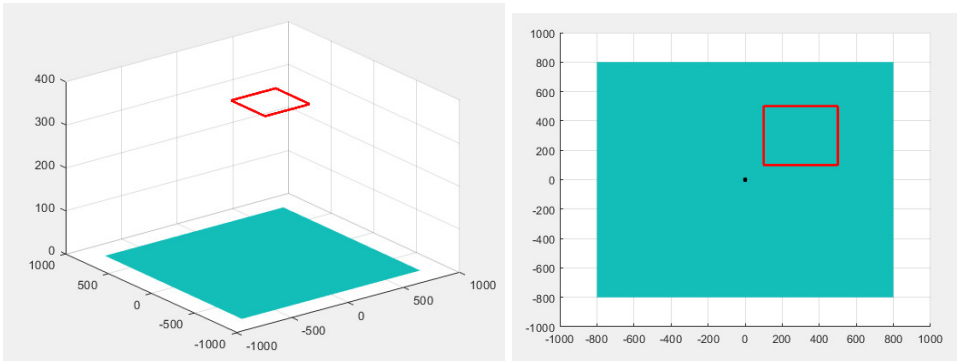


Figure 4.10: Tool tip trajectory



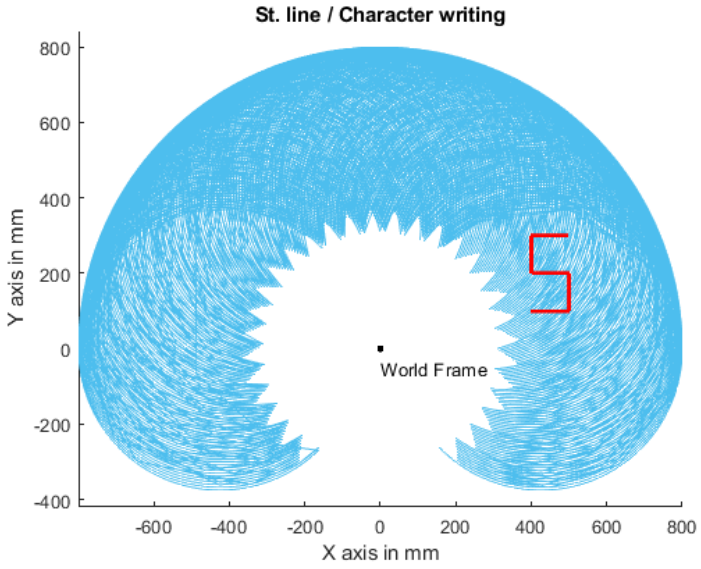
(a) Isometric view (b) Top view

Figure 4.11: Tool tip trajectory

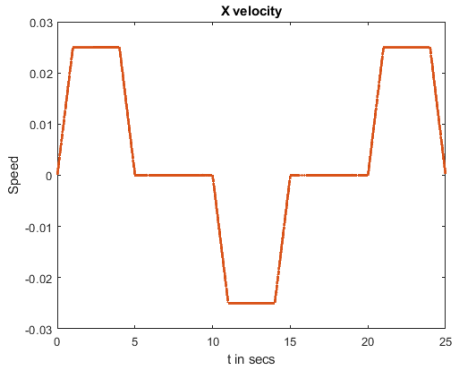
4.3.3 Character Writing

Approximate alphabetical characters can also be obtained using the same method. In Figure 4.12(a), we have traversed the shape of 'S' marked by red color. Five straight lines were used to make up the letter. In practical, a pen/marker could be attached to the end effector in order to draw it on paper base. It is also important that we trace the letter

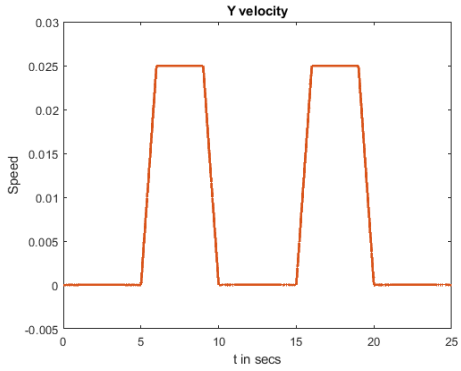
'S' within workspace only. Blue color region in the figure shows the workspace of the arm. In Figure 4.12(b) and (c), velocities of the tool tip in X and Y directions are plotted with total time period = 25s i.e. to traverse 5 edges in 'S'. Same profiles are observed as what we had expected from Figure 4.9(a).



(a) 'S' letter traversed



(b) Velocity profile in X direction



(c) Velocity profile in Y direction

Figure 4.12: Writing character 'S'

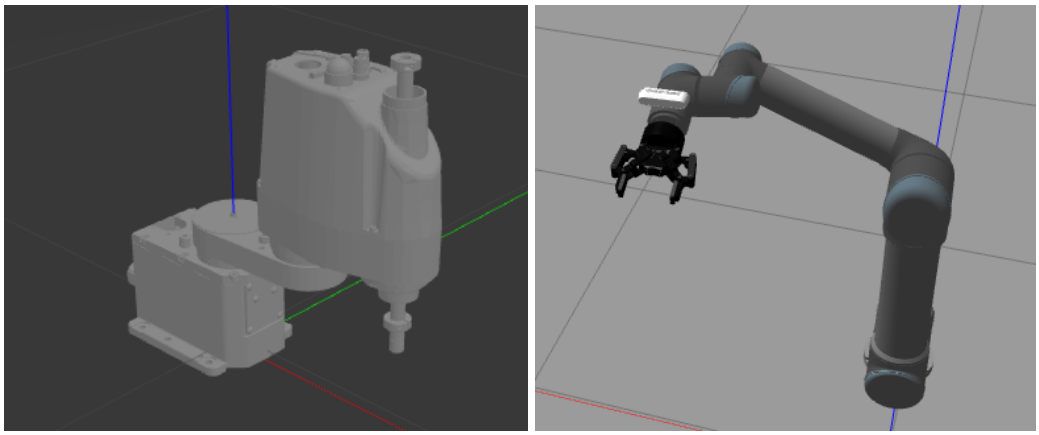
4.4 ROS based control and simulation

This point onwards, all simulations were performed using ROS and Gazebo. In order to simulate, we first need to setup the necessary files first and then write scripts to control the arm.

4.4.1 Setup

URDF

The Unified Robotic Description Format (URDF) is an XML file format that describes robot's physical description to ROS. These files are needed to understand and be able to simulate situations with the robot. The URDF file for SCARA was written by us whereas for UR5 arm we acquired it through its Github repository[6]. As SCARA is a general class of robots, we used Epson SCARA G3-251S[7] as the reference model. Models generated are shown in Figure 4.13.



(a) G3-251S in Gazebo (b) UR5 in Gazebo

Figure 4.13: Robot arms seen in Gazebo simulator

Setting up MoveIt!

MoveIt![8] is a very popular motion planning framework.. It is free for industrial, commercial, and research use. It contains software for manipulation, motion planning, kinematics, collision checking etc. To get started with it, we had to go through the documentation and tutorials available[9]. Figure 4.14 shows it's high-level system architecture.

A planning scene consists of the information needed to compute motion plans which includes current state of the robot and representation of the robot and the world. It uses Octomap to maintain the occupancy map of the environment. The kinematics

[6] https://github.com/ros-industrial/universal_robot

[7] https://www.epson.de/en/products/robot/epson-scara-g3-251s

[8] https://moveit.ros.org/

[9] http://docs.ros.org/en/melodic/api/moveit_tutorials/html/doc/getting_started/getting_started.html

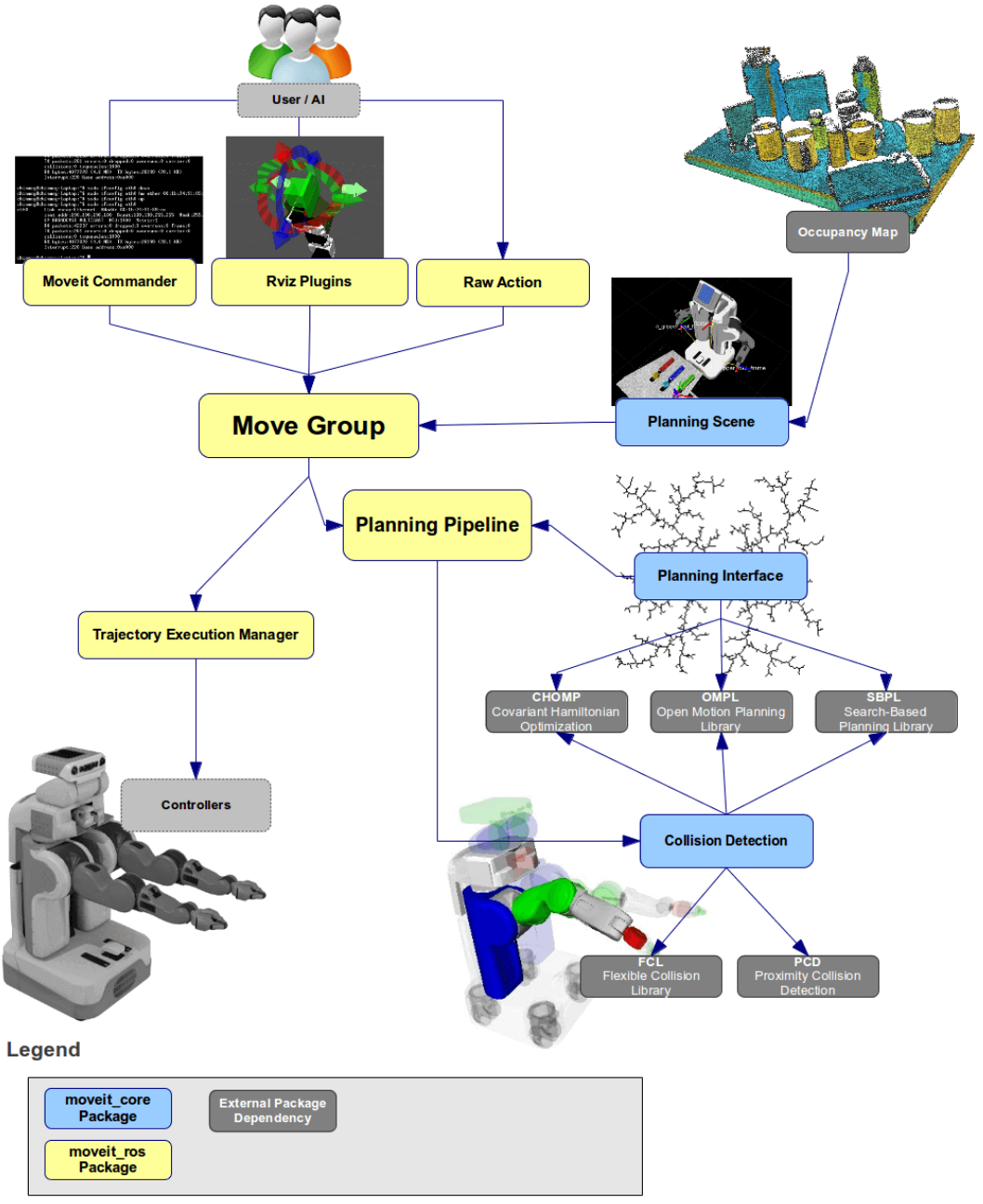


Figure 4.14: MoveIt! Quick High Level Diagram

solver is the module which is going to solve the kinematics of the robot whether it be direct or inverse kinematics. MoveIt! uses a plugin type of interface for the kinematics solver where the desired kinematics solver plugin can be selected while making the package. Some of the various plugins it offers include KDL, IKFAST and TRAK-IK. The one which we have used for our project is KDL. It also allows for custom plugins. Collision checking in MoveIt! is configured inside a Planning Scene. Fortunately, MoveIt is setup so that users never really have to worry about how collision checking is happening. Motion planning is the process of breaking down a desired movement task into discrete motions that doesn't violate the various constraints specified and possibly optimize some aspect of the movement. MoveIt! provides us with various planners like OMPL and CHOMP. The planners can be used as single entity or can be pipelined to improve the plan made. Default planner OMPL was sufficent for our project. Additional information on how MoveIt! works can be found here[10].

MoveIt! provides a GUI interface called MoveIt! Setup Assistant (Figure 4.15) to generate all the configuration files needed. After configuring all the options, we click on generate package. This will generate and write a set of launch and config files into the directory of our choosing.

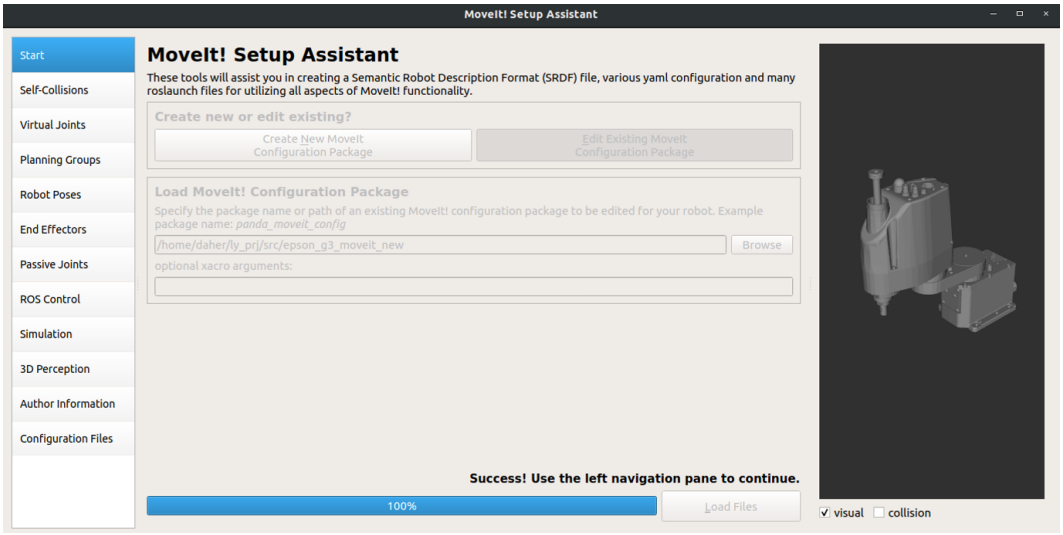


Figure 4.15: MoveIt! Setup Assistant

Scripts

MoveIt! provides C++ and Python APIs to perform actions like setting joint or pose goals, creating motion plans, moving the robot, adding objects into the environment

[10] https://moveit.ros.org/documentation/concepts/

and attaching/detaching objects from the robot. We wrote several scripts to perform applications with both the robot arms. They are presented below.

4.4.2 Simulations

SCARA

As discussed in theoretical analysis, robot arm can be actuated to follow a certain trajectory in cartesian space. With SCARA robot, we first traversed random points, then we made the tool tip traverse 'H' character as shown in Figure 4.16 and Figure 4.17. Since pick and place was already simulated in section 4.3.1, we did not do it here.

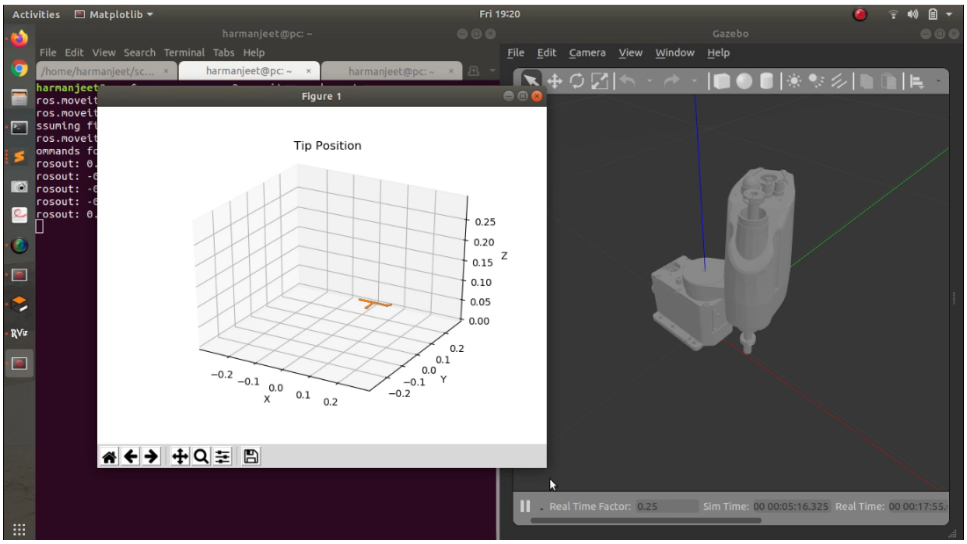
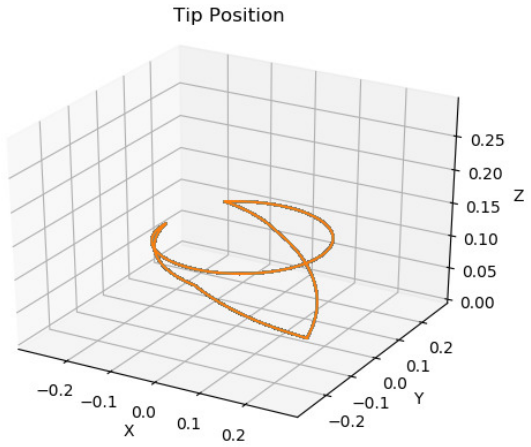
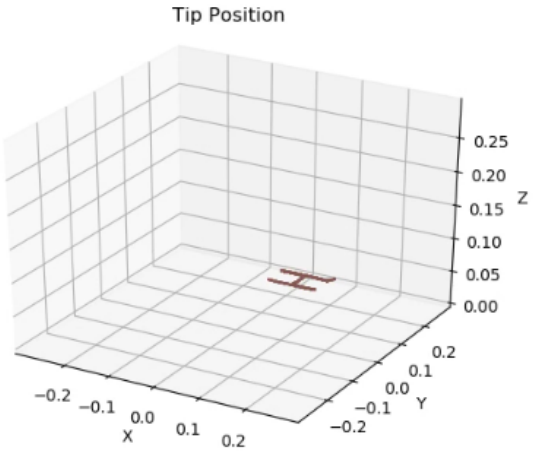


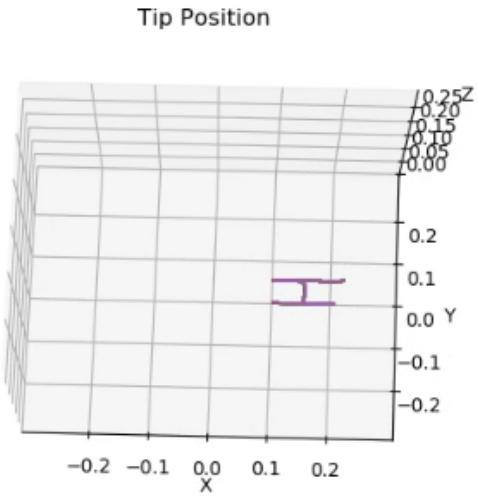
Figure 4.16: SCARA- Simulation being performed



(a) Random points



(b) 'H' Isometric

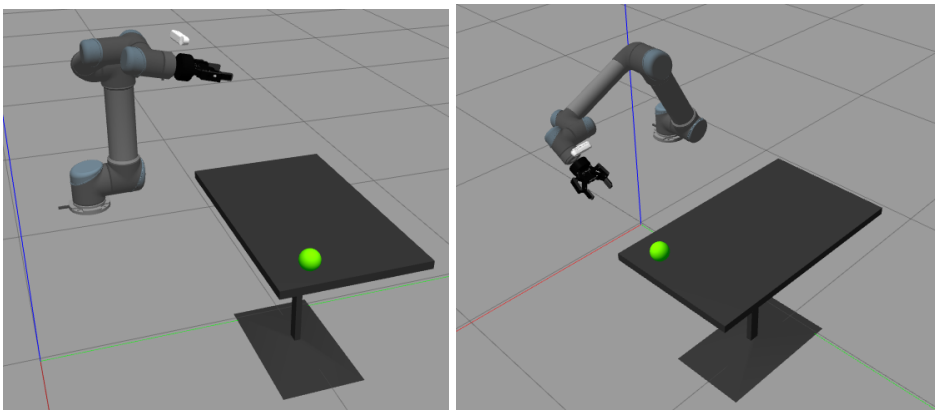


(c) 'H' Top view

Figure 4.17: Tool tip position from simulations performed in Gazebo simulator

UR5

Here, we did not do pick and place alone. We thought of picking up a rolling ball from the table. The setup is shown in Figure 4.18.



(a) (b)

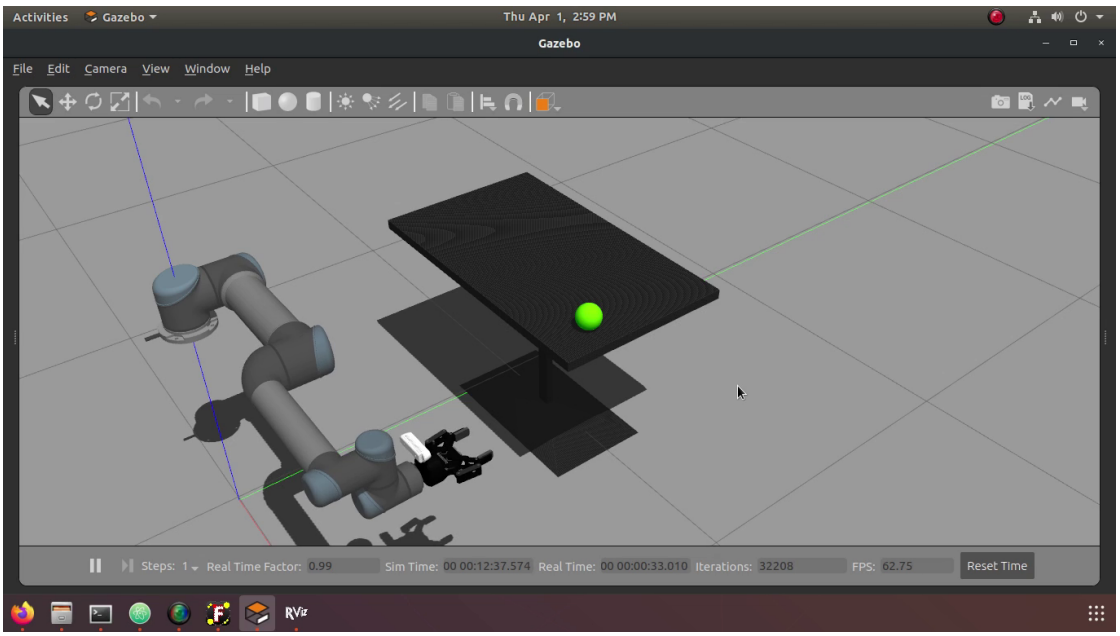
Figure 4.18: UR5: Picking up a ball rolling on table setup

A depth camera is attached to the robotic arm. A computer vision package called *find_object_2d*[11] was used. It helped to determine the position of the ball with respect to the arm's base frame. Figure 4.19 shows the GUI of the find_object_2d package after it has been configured. The left column shows the template that the program is going to be looking for in the feed it gets from the camera. Multiple template images were given to improve the accuracy and chances of detection. The center column consists of the live camera feed. The right column consists of the various parameters which can be adjusted like the various descriptors, detectors, their thresholds, etc.

Initially the arms follows the ball and then it starts coming closer to the ball. The ball is being approached at 45 degrees angle. When the arm's fingers get around the ball, it closes the gripper and grabs the ball and returns to home position. The simulation snapshots are attached in Figure 4.20.

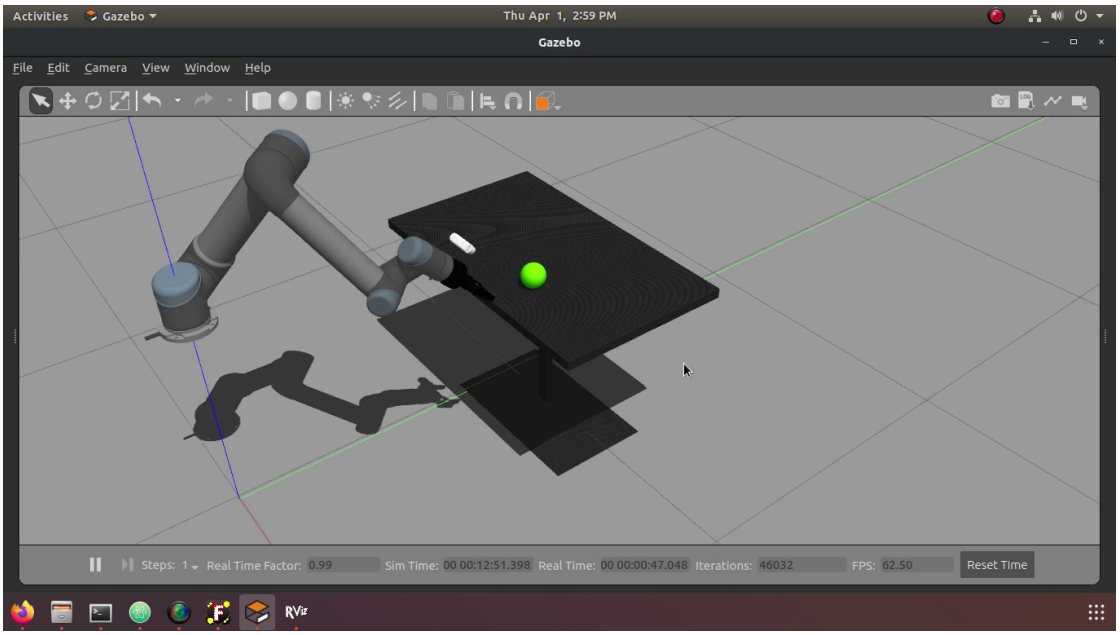
[11] `https://wiki.ros.org/find_object_2d`

Figure 4.19: Find_Object GUI

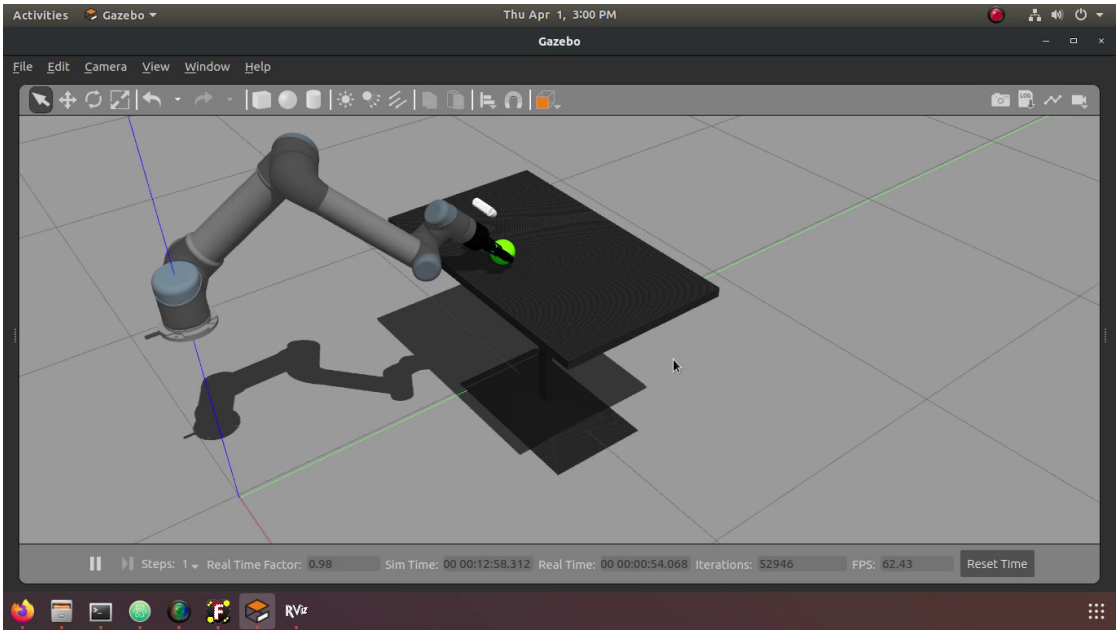


(a)

Figure 4.20: UR5: Picking up a ball rolling on table simulation

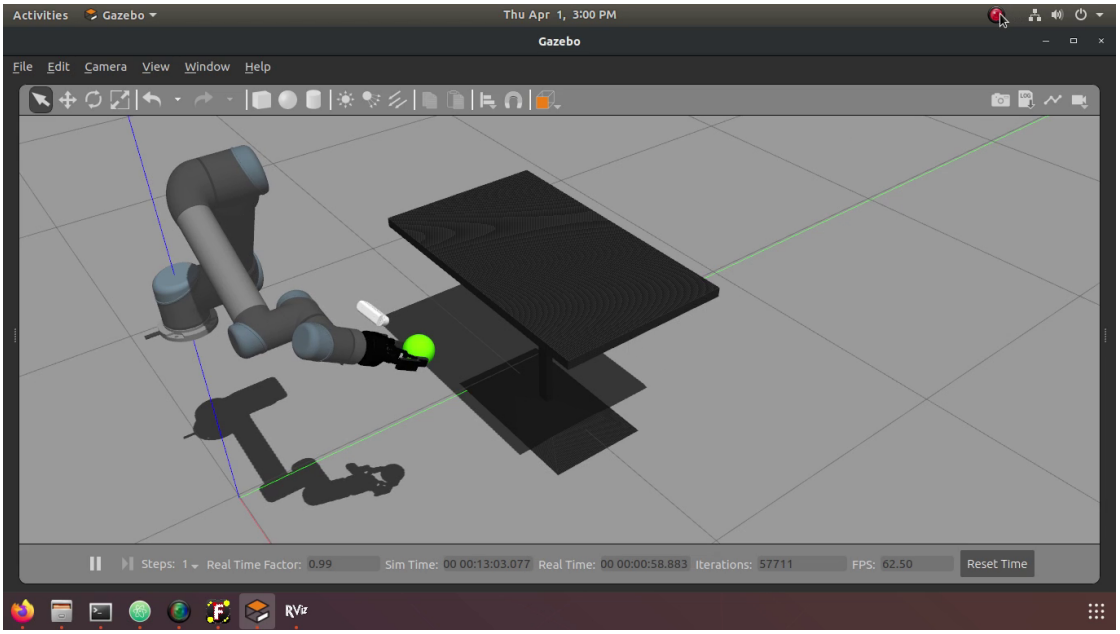


(b)



(c)

Figure 4.20: UR5: Picking up a ball rolling on table simulation (cont.)



(d)

Figure 4.20: UR5: Picking up a ball rolling on table simulation (cont.)

4.5 Results and Discussion

The analytic direct and inverse kinematic equations of SCARA were verified using the Simscape model created. For UR5, excel sheet was used to visualize the direct kinematics. KDL plugin was used as the kinematics solver for simulations in ROS. We plotted SCARA's horizontal cross-section of work envelope for two different joint ranges. Bigger joint ranges covered more area, as expected. The workspaces can be extended in the z-direction easily as there's only prismatic joint affecting it. UR5 robot having six revolute joints has a spherical work envelope. It is not the dexterous envelope. Several orientations are not achievable at some points.

We successfully simulated pick and place operation with SCARA in MATLAB/Simulink. Here the object was picked and place at different elevation levels. This application was completed using point to point motion control and thus the kinematics were further verified. To reduce the time taken for the whole process the prismatic joint was actuated during the radial motion itself. But this can cause collision with environment. One can actuate the prismatic joint individually without the radial movement to solve it.

The SCARA robot was further commanded to follow a straight line in tool configuration space. Using multiple such lines, a rectangular shape and character 'S' was also traversed by the tool tip. The only undesired element is the presence of sharp points in velocity profile as this will cause jerks when implemented in a physical setup. These sharp points can be smoothened out using parabolic blending which will solve the issue.

At this point we shifted to simulations in ROS. Here, we first configured both the arms to use them with MoveIt!. We then wrote several scripts to control the arm in Gazebo.

With SCARA, we first traversed random points. Then we approximately traversed the letter 'H' using six knot points. The tool tip passed the desired points accurately but the motion observed for traversal between points was found out to be not straight which caused slight deformities in the character written. To solve this, more knot points could be added in an optimal manner.

With UR5, using depth camera we performed object's pose estimation. The rolling ball was followed by the UR5 arm for certain period and then was picked up with a firm grip. The arm was able to come back to start position successfully without dropping the ball. Due to computing power limitations, the camera image update rate and the rate at which the controller operated, could not be increased beyond a certain limit which caused a slight delay in ball following.

Chapter 5

Conclusions and Future Work

Understanding the fundamental concepts was the up front task that was completed by us. This led us to complete the theoretical analysis of both the arms. We were able to analyse and control them. We learned how to use tools like MATLAB/Simulink, ROS, Gazebo and libraries like MoveIt!, FindObject for different tasks. Using these tools, we completed a number of simulations as discussed above. Both the arms have acted more like a tool for testing, verification and visualization of the theory we had learned. It is also important to note that the control strategies that we have discussed will require smoothening before hardware deployment to decrease rough and jerky movements.

This project has opened doors for us to study the recent advancements in the subject. This includes motion planning, collision avoidance, generic kinematics algorithms and more. We used these features through MoveIt!, but the next task will be to understand how these algorithms work, how are they coded and how are they implemented in hardware. The dynamic model of the robot arm can be studied for precise control of high speed motion. A great deal of learning is left on how to use the capabilities of simulation softwares to the fullest. The open source nature of ROS motivates us to learn more and then share our work with the community.

Finally, we shall always look up to implementing the concepts in hardware when possible.

Appendix A

Simulation Video Links

Simulink - SCARA Pick and Place	https://youtu.be/lC4x-LoYQdg
Simulink - SCARA traversing 'S'	https://youtu.be/CsPLOTyOAQw
ROS/Gazebo - SCARA traversing 'H'	https://youtu.be/aVlcPpxdjcQ
ROS/Gazebo - UR5 Picking up a ball	https://youtu.be/BM52mtVZ3e8

References

Andersen, R. S., 2018, "Kinematics of a ur5," *Aalborg University: Aalborg, Denmark*

Beeson, P., and Ames, B., 2015, "Trac-ik: An open-source library for improved solving of generic inverse kinematics," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)* (IEEE). pp. 928–935.

Construct Sim, 2021, `https://www.theconstructsim.com/`

Craig, J. J., 2009, *Introduction to robotics: mechanics and control, 3/E* (Pearson Education India).

De Gier, M., 2015, "Control of a robotic arm: Application to on-surface 3d-printing,"

Gasparetto, A., and Scalera, L., 2019, "A brief history of industrial robotics in the 20th century," *Advances in Historical Studies* **8**, 24–35.

Hawkins, K. P., 2013, *Analytic inverse kinematics for the universal robots UR-5/UR-10 arms*, Tech. Rep. (Georgia Institute of Technology).

Khokar, K., Beeson, P., and Burridge, R., 2015, "Implementation of kdl inverse kinematics routine on the atlas humanoid robot," *Procedia Computer Science* **46**, 1441–1448.

Labbé, M., 2011, "Find-Object," `http://introlab.github.io/find-object`

MoveIt! Concepts, 2021, `https://moveit.ros.org/documentation/concepts/`

NASA Mars Perserverance rover, 2020, `https://mars.nasa.gov/mars2020/spacecraft/rover/arm/`

O'Kane, J. M., 2014, "A gentle introduction to ros,"

Pyo, Y., Cho, H., Jung, R., and Lim, T., 2017, "Ros robot programming," *ROBOTIS, December*

Remote Manipulator System (Canadarm2), 2020, https://www.nasa.gov/mission_pages/station/structure/elements/remote-manipulator-system-canadarm2/

Santoni, A., 2016, "A ros-based workspace control and trajectory planner for a seven degrees of freedom robotic arm,"

Schilling, R. J., 1996, *Fundamentals of robotics: analysis and control* (Simon & Schuster Trade).

Taylor, R. H., 1979, "Planning and execution of straight line manipulator trajectories," *IBM Journal of Research and Development* **23**, 424–436.

Universal Robots, 2021, https://www.universal-robots.com/

Acknowledgements

We would like to express our gratitude to supervisor Prof. Annu Abraham for always backing and guiding us throughout the year.

Saeesh Aher, Harmanjeet Singh Bilkhu, Dhairya Maisheri
KJSCE
14 June 2021

Analysis and Simulation of Robotic Arms:SCARA & UR5

ORIGINALITY REPORT

9%	9%	3%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.scribd.com Internet Source	2%
2	core.ac.uk Internet Source	2%
3	Submitted to Indian Institute of Technology, Bombay Student Paper	2%
4	www.universal-robots.com Internet Source	1%
5	en.unionpedia.org Internet Source	1%
6	www.scirp.org Internet Source	1%
7	buildmedia.readthedocs.org Internet Source	1%
8	Submitted to Sogang University Student Paper	1%
9	www.mathworks.com Internet Source	1%

Exclude quotes	On	Exclude matches	< 1%
Exclude bibliography	On		