



Algorithms for Object Palletizing using Robotic Arm

Project Report

**Submitted in partial fulfillment of the requirement
for the award of the degree of
Bachelor of Technology in Electronics Engineering**

by

Maha Zakir Khan 19ELB056

Shaista Tabrez 19ELB057

Supervisor

DR. Mohd Wajid

**DEPARTMENT OF ELECTRONICS ENGINEERING
ZAKIR HUSAIN COLLEGE OF ENGINEERING & TECHNOLOGY
ALIGARH MUSLIM UNIVERSITY ALIGARH
ALIGARH-202002 (INDIA)
May 2023**

STUDENTS' DECLARATION

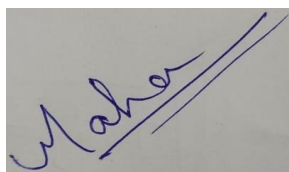
I hereby certify that the work which is being presented in this project report entitled “**Algorithms for Object Palletizing using Robotic Arm**” in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology and submitted in the Department of Electronics Engineering of the Zakir Husain College of Engineering & Technology, Aligarh Muslim University, Aligarh is an authentic record of my own work carried out during the final year of B.Tech. under the guidance of **Dr. Mohd Wajid**, Department of Electronics Engineering, Aligarh Muslim University, Aligarh.

We have also checked the plagiarism of this report with the Turnitin software from Maulana Azad Library of AMU with the following details:

- Submission Date for plagiarism check: _09-May-2023_
- Submission ID for plagiarism check: ___2088360278___

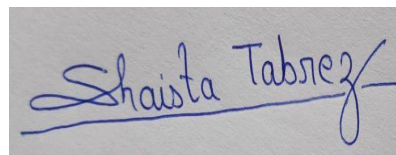
Note: These above details shall be available on the footer of the first page of the plagiarism report

- Similarity Index: _____9_____percent



(Maha Zakir Khan)

Faculty Number: 19ELB056



(Shaista Tabrez)

Faculty Number: 19ELB057

The above statement made by students seems to be correct to the best of my knowledge.

(Dr. Mohd Wajid)

Project Guide

Date: 10-May-2023

ABSTRACT

Industries are using robots more and more frequently for automation activities, but using robots at home or at work to work with humans on everyday tasks is still considered to be a vision of the future. This is because operating a system that can carry out general tasks involves an immense amount of complexity. Using a 3D printed 5 Degree of Freedom Robotic Arm, we are exploring robot kinematics motion planning to make it easier to automate the task of randomly selecting scattered coloured cubes that are 4x4cm identical cubes on a 23.5x23.5 cm platform and placing them in an organized pattern at a specific location on the side (placement station) of the platform.

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest gratitude to my supervisor Dr. Mohd Wajd for his inspiration and encouragement in the successful completion of this work. He has always been cordial, attentive, responsible and supportive throughout all the highs and lows during the journey of my project work. He invested a lot of his time for our project, giving valuable inputs and took great pains to see us through. I have learnt a lot from him and I humbly acknowledge a lifetime deep gratitude to him. He devoted a lot of time and patience to the reading and correction of this report. I am thankful to Prof. Tahira Prveen and Prof. Omar Farooq for being present during various presentations of my project work and providing me with their valuable comments and suggestions. I am also thankful to Prof. Ekram Khan, Chairman of ECE department for his constant support and motivation for completing our project work.

Date: 10-May-2023

Place: Aligarh

(Dr. Mohd Wajid)

TABLE OF CONTENTS

	Page No
Candidate's Declaration	
Abstract	i
Acknowledgements	vi
Contents	vii
List of Figures	xi
List of Tables	xvii
List of Symbols	xix
List of Abbreviations	xxiii
 CHAPTER 1: Introduction	
 CHAPTER 2: Modeling	
2.1 Hardware description	14
2.2 Software Used	14
 CHAPTER 3: Kinematics of Robot	
3.1 Forward Modelling of the 5 DOF Robotic Arm	18
3.2 DH Forward Kinematics model of manipulator	19
3.3 Managing the orientation of the end-effector	21
3.4 Inverse Kinematics (IK)	23
 CHAPTER 4: Simulation	

CHAPTER 5: Trajectory Planning

5.1	What is Motion Planning	29
5.2	Common types of motion planning	29
5.3	Types of trajectory	31
5.4	Trajectory generation techniques	34
5.4.1	Trapezoidal Trajectory	34
5.4.2	Polynomial Trajectory	35
5.5	Rational Interpolation	36

CHAPTER 6: Perception

6.1	Coordinate estimation of target T	39
6.2	Perception System Design	40
6.3	Estimate coordinate of target T	42

CHAPTER 7: Hardware Implementation

7.1	Process of Pelletizing	44
-----	------------------------	----

CHAPTER 8: Conclusion and Further Scope

7.1	Conclusion	47
7.2	Scope of future research	47

LIST OF FIGURES

Figure No.	Caption	Page No.
Fig. 2.1	An illustration of the 5-DOF robotic arm in 3D in SolidWorks.	15
Fig. 2.2	URDF model of the robot arm (Red: x-axis, Green: y-axis, Blue: z-axis).	15
Fig. 2.3	Link Lengths in centimeters.	16
Fig. 3.1	Kinematics Block Diagram.	17
Fig. 3.2	Kinematic and frame representation.	19
Fig. 3.3	Euler convention rotations $R_x(\gamma)$, $R_y(\beta)$ and $R_z(\alpha)$	22
Fig. 4.1	Rigid Body Tree of the manipulator	25
Fig. 4.2	Forward Kinematics Block	26
Fig. 4.3	Complete Forward Kinematics Implementation	27
Fig. 4.4	Plot 1 Y axis-Amplitude X axis- Time in seconds	27
Fig. 5.1	Autonomous System Workflow	28
Fig. 5.2	Task Space Vs Joint Space Trajectory	31
Fig. 5.3	Joint Space Trajectory and Waypoints	32
Fig. 5.4	Joint Space Velocities	33
Fig. 5.5	Joint Space and Task Space Simulation	33
Fig 5.6.	Trapezoidal profile of manipulator arm	34
Fig. 5.7.	Basic cubic polynomial Trajectory Profile of Manipulator	35
Fig. 5.8.	Rotating end-effector using SLERP method	37
Fig. 6.1.	Perception for intelligent robot	38
Fig. 6.2.	Platform for detecting cubes (S_x, S_y), Placement Station for stacking/sorting the cubes (P_x, P_y)	39
Fig. 6.3.	Perception Workflow	40
Fig. 6.4.	A simplified model of robotic arm for intuitive estimation of the robot position	41

Fig. 6.5.	Placement Station detection (a) Original Image (b) Gaussian Blur (c) Canny Edge Detector (d) Contour Detection [Image size: 620x480 pixels]	41
Fig. 6.6.	Placement Station (a) Cropped Platform Image (Distortion removed) (b) Canny edge detection (c) Placement Station Detection [Image size: 500x500 pixels]	42
Fig. 6.7	Platform for detecting cubes (S_x, S_y), Placement Station for stacking/sorting the cubes (P_x, P_y)	42
Fig. 6.8.	Colour Masking (a) Original Image (b) Mask (c) Result (hsv value for red cube $h=0$ $s=108$ $v=4$)	43
Fig. 7.1.	Work flow of pick place and stack operation	44
Fig. 7.2.	5 DOF Robotic Arm performing task of palletization	45

LIST OF TABLES

Table No.	Title	Page No.
Table 1	DH Parameter of Forward Kinematics Model	19
Table 2	Task Space Vs Joint Space Trajectories.	32

LIST OF SYMBOLS

3D	Three Dimensional
T	Homogeneous Transformation Matrix
i	Joint Sequence
a_i	Link length
α_i	Link Twist
d_i	Link offset
θ_i	Joint Angle
O	Origin
X,Y,Z	Coordinate Axis
r_{ij}	Orientation of end-effector
P_x, P_y, P_z	Coordinate of the end-effector
C_i	$\cos(\theta_i)$
S_i	$\sin(\theta_i)$
α, β, γ	Euler Angle Convention
Q()	Joint Coordinate
p_x, p_y, p_z	Gripper Position
$f1(Q)$	Euclidean distance of the end-effector to the target position
$f2(Q)$	Euclidean distance of the end-effector to the target orientation
$\alpha_t, \beta_t, \gamma_t$	Target Orientation

LIST OF ABBREVIATIONS

DOF	Degree Of Freedom
URDF	Unified Robotics Description Format
IK	Inverse Kinematics
DH	Denavit-Hartenberg
RBT	Rigid Body Tree
UAV	Unmanned Aerial Vehicles
UGV	Unmanned Ground Vehicle
RRT	Rapidly Exploring Random Tree
LIDAR	Light Detection and Ranging
PID	Proportional Integral Derivative
Slerp	Spherical Linear Interpolation
IP	Internet Protocol

Chapter 1

Introduction

Palletizing refers to the task of arranging similar objects like a carton or a sack and arranging it in a defined pattern. Industries are witnessing a rapid rise in the use of robots for automation tasks, however robots at home or robots at the workplace assisting humans side by side in day-to-day tasks is still seen as a concept of the future. This is due to the immense complexities that are involved in the execution of such a system capable of performing general tasks. We plan on devising a way to introduce robot in non-industrial settings helping people in common everyday problems like sorting and stacking disorganized objects in homes, workplaces or industrial settings where the size/shape and location of manufactured products keeps on changing at frequent intervals and designing a completely new manipulator would be time consuming and expensive.

A robot arm can also be referred to as a Manipulator, which is created from a sequence of links and joint combinations. The gripper of the robot can also be referred to as End-Effector. We will be using both terms interchangeably from now on. In order to implement the algorithm, we first need to do forward modeling of the robotic arm. While implementing and reviewing literature we have realized one of the toughest robotics problems is inverse kinematics(IK), instead of forward kinematics, where the end-tip position is sought given the pose or joint, the issue is determining the best pose for a manipulator given the location of the end-tip effector has multiple solutions. This position is typically expressed as a point in a coordinate system, such as the Cartesian system with the coordinates xx , yy , and zz . The collection of joint variables that

represent the length of extension (in prismatic joints) or the angle of bending or twisting (in revolute joints) can also be used to express the pose of the manipulator.

However, Inverse Kinematics is challenging, though, because there are numerous potential fixes. One might intuitively understand that a robotic arm would be able to pass through a particular spot in a variety of ways. The same holds true if you touch the table and move your arm without changing the angle at which you are touching the table. Additionally, calculating these positions can be very challenging. For manipulators with three degrees of freedom (DOF), straightforward solutions can be found, but when five or more DOF are involved, complex algebraic issues might arise. To avoid calculation of IK at every point on the trajectory and reduce computational costs we will be relying on joint space trajectory generation.

Objectives

- Build and test simulation model for kinematic analysis
- To explore Trajectory Planning Algorithm
- Optimize the algorithm for palletizing tasks and test it on the simulation model.
- Detecting and tracking position of objects to be sorted and stacked (small cubes/cuboids in our case)
- Picking up, carrying and placing objects using the simulation model.
- Deployment on hardware and testing in a structured environment.

Chapter 2

Modeling

1.1 Hardware Description:

A robotic arm that was manufactured using a 3D printer for educational purposes that has five degrees of freedom, except the gripper joint, as shown in "Fig. 1" is used. While the two wrist roll and wrist pitch joints are moved by a smaller micro-servo, the MG90S, the first three rotational joints, starting from the base, are moved by an MG996R servo motor. Another MG90S servo, not shown on the model, is used to control the gripper's grabbing. An Arduino board is used to control the system. Such a robotic arm is inexpensive and has many potential educational uses. It enables us to comprehend the fundamental problems with pacifying the actions of robotic arms, which are shared by industrial applications for arms.

1.2 Software Used:

SolidWorks is used to model the robot for 3D printing as well as generate an XML file for extracting a rigid body tree to be used for MATLAB Simscape multibody for studying forwards kinematics, IK as well as trajectories accounting for the inertia considerations.

For a simpler rendition for quick implementation, we are going to use a URDF model to simulate the target position of the robot gripper (End Effector) to be achieved

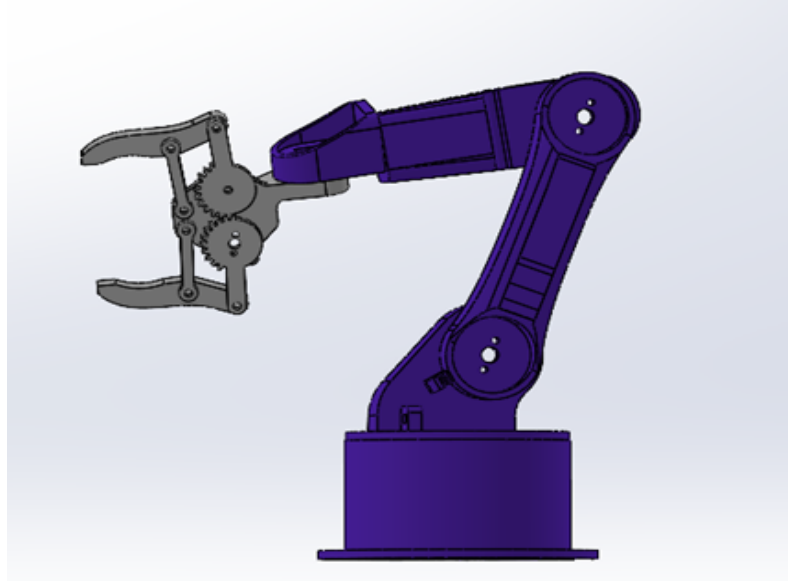


Fig 2.1. An illustration of the 5-DOF robotic arm in 3D in SolidWorks

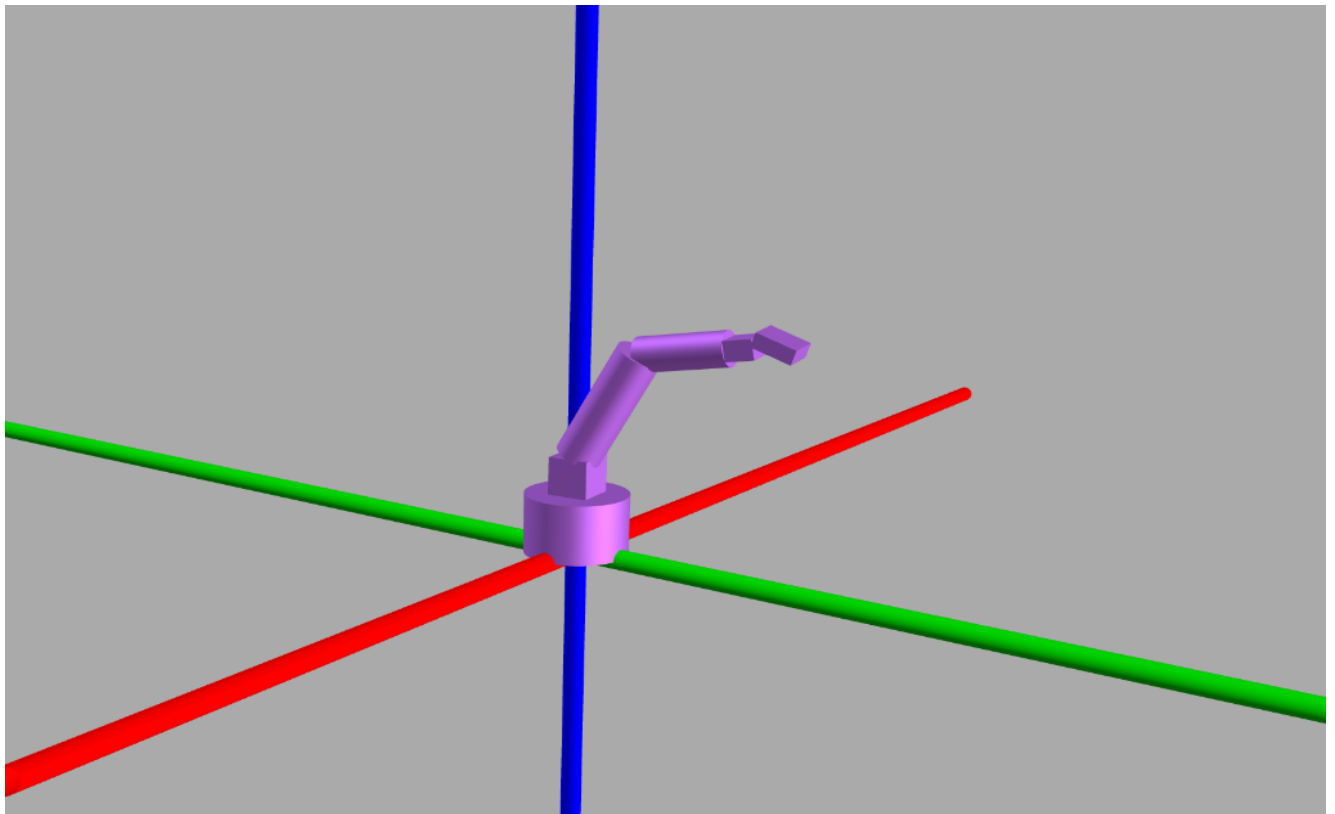


Fig 2.2. URDF model of the robot Arm (Red: x-axis, Green: y-axis, Blue: z-axis)

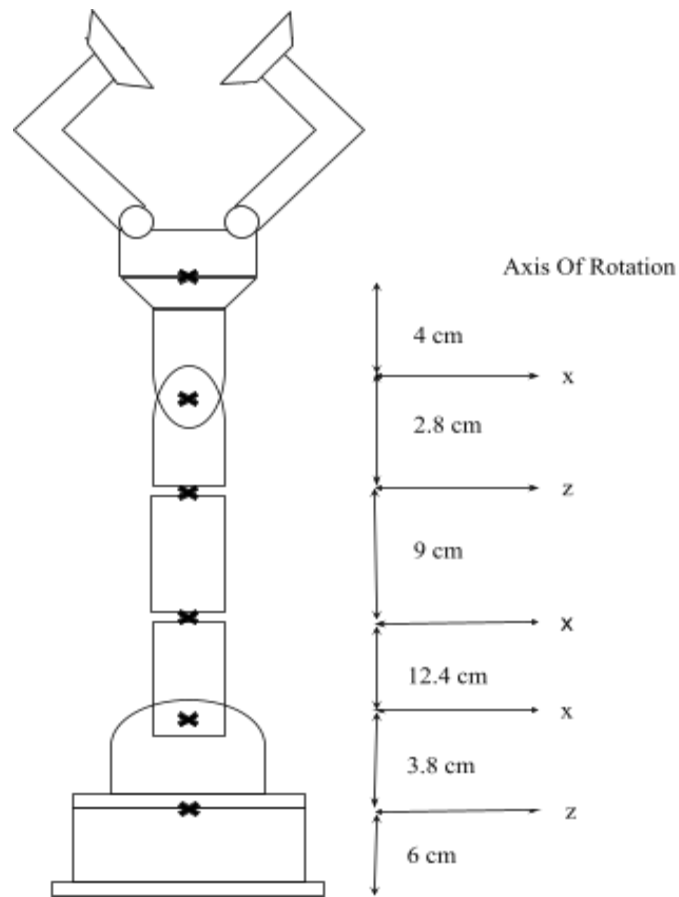


Fig 2.3. Link Lengths in centimeters

Chapter 3

Kinematics of Robot

Kinematics is the study of how a robot's joint coordinates and spatial configuration interact. The movement of the robot under various stresses and torques is not taken into account; just the immediate values of the robot's coordinates are considered. Kinematics essentially describes how a robot moves without taking into account what causes it to move. It deals with the investigation of the variables related to location, velocity, acceleration, and higher derivatives of the position variables. Kinematics can help in achieving accurate results in problems such as how robots will move from one point to another, how to overcome colliding with the obstacles in the path, and position of the gripper to properly grasp objects etc. Forward and inverse kinematics are the two main divisions of kinematics.

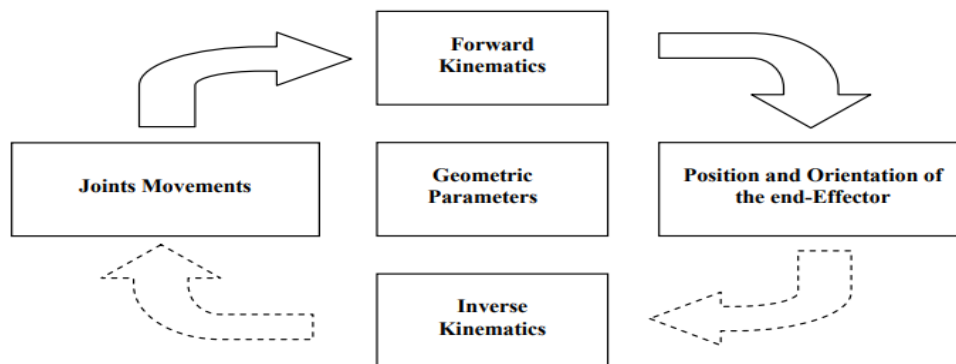


Fig 3.1. Kinematics Block Diagram

3.1. Forward Modelling of the 5 DOF Robotic Arm

Forward Kinematics also known as Direct Kinematics basically refers to the process of obtaining position and velocity of the gripper, provided the robot's joint parameters are known. Joint variables apply to the angles between the links in revolute or rotational joints, whereas they apply to the link extension in prismatic or sliding joints. The simplest and most popular method for determining the expression for the forward kinematics of the 5-DOF robotic arm is Denavit-Hartenberg analysis. To determine how the joints and links should be positioned and oriented with relation to one another four parameters are required known as DH parameters. These four parameters are link length, link offset, link twist and joint angle.

Link Length a_i : This term describes the link's length between two joints.

Link Offset d_i : This term describes the distance between the joints on either side of the link. The offset will be zero if the link is straight and connects the two links. The joints will probably be offset if the link has a bend or kink in it.

Link Twist α_i : This term describes the angle formed when comparing the joints at the beginning and end of a connection. The twist of the connection is zero if the axes of the two joints are parallel.

Joint Angle θ_i : It refers to the angle between two joints along a link.

Table 1 lists the DH arm parameters, and "Fig. 2b" shows the displacements and rotations of the related frames.

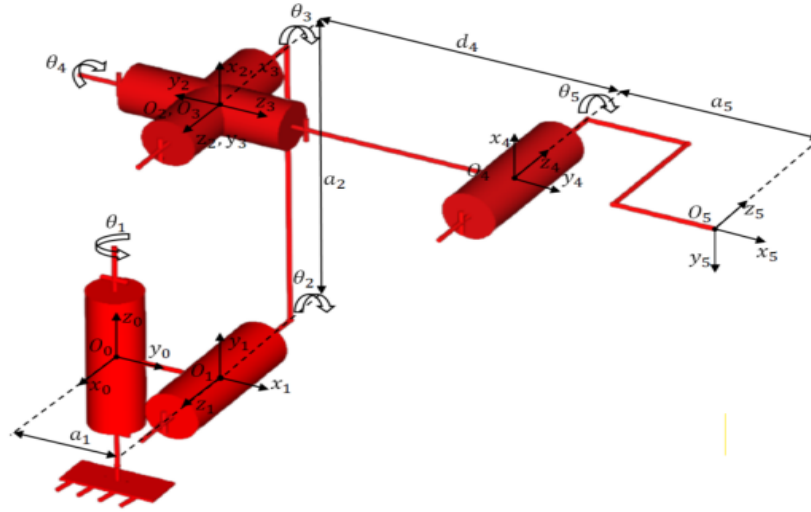


Fig 3.2. Kinematic and frame representations

Table 1. 5 DOF Manipulator Denavit-Hartenberg Parameters

Frame (i)	a_i (cm)	α_i (°)	d_i (cm)	θ_i (°)
1	1.374	90	0	[0, 180]
2	12	0	0	[0, 170]
3	0	90	0	[0, 135]
5	0	90	11.965	[0, 180]
5	9.887	0	0	[0, 180]

3.2. Denavit–Hartenberg Forward Kinematics Model of the Manipulator

In order to define the forward kinematics model of a serial robot, the Denavit-Hartenberg (DH) convention uses a generic transformation matrix to model the position and orientation between two sequential frames (i) and (i-1), (T^{i-1}_i). Table 1 shows the robotic arm's DH parameters. In this table, "i" stands for the joint sequence, " a_i " for the distance between " O_{i-1} " and " O_i " along the x-axis, " α_i " for the angle between " Z_i " and " X_{i-1} ", " d_i " for the distance from " X_{i-1} " to " X_i " along

Z_{i-1} . It is possible to use basic homogeneous transformation operations in this analysis once the coordinate transformation between two frames, where the position and orientation are fixed with respect to one another, has been determined.

A 4x4 homogeneous matrix (T_5^0), produced by multiplying the aforementioned matrices as stated in (1), is used to express how the gripper of the 5-DOF manipulator is positioned and orientated.

$$T_5^0 = T_1^0 \cdot T_2^1 \cdot T_3^2 \cdot T_4^3 \cdot T_5^4 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

where,

T_i^{i-1} : It refers to the homogeneous transformation matrix, which explains how frame (i) changes into frame (i-1).

r_{ij} : It describes how the gripper is oriented.

p_i : It provides the end-effector's coordinates in relation to the base frame, and it is finally expressed by equations 2(i, ii, iii).

$$P_x = a_5 C_4 C_5 (C_1 C_2 C_3 - C_1 S_2 S_3) + a_5 C_4 C_5 S_1 + (a_5 S_5 + d_4 (C_1 C_2 S_3 + C_1 S_2 C_3)) + (a_2 C_1 C_2 + a_1 C_1) \quad 2(i)$$

$$P_y = a_5 C_4 C_5 (S_1 C_2 C_3 - S_1 S_2 S_3) - a_5 C_4 C_5 \cdot C_1 + (a_5 S_5 + d_4 (S_1 C_2 S_3 + S_1 C_2 C_3)) + (a_2 S_1 C_2 + a_1 S_1) \quad 2(ii)$$

$$P_z = a_5 C_4 C_5 (S_2 C_3 + C_2 S_3) + (a_5 S_5 + d_4 (S_2 S_3 - C_2 C_3)) + a_2 S_2 \quad 2(iii)$$

S_i represents $\sin(\theta_i)$, whereas C_i stands for $\cos(\theta_i)$. The Denavit- Hartenberg, or DH, parameters are denoted by the terms (a_i) and (d_i) , which are listed in Table 1. The end-effector's location and orientations can be retrieved using this Denavit-Hartenberg kinematic modeling technique. Euler angles, which are frequently employed in managing the orientation of robots, must be translated from orientations. The next paragraph goes into more depth about this transition.

3.3 Managing the Orientation of the End-Effector:

Euler angles convention (α, β, γ) is a standard convention used for expressing the orientation of the robotic arm which is represented in fig.3. They are described as three (chained) rotations with respect to the coordinate frame's three main axes. The Euler convention represented in Fig.3, specifies orientations of the arm as follows:

- First frame is initially rotated by an angle γ around $-X_0$, of frame 0.
- Then the second frame is rotated by an angle β around $-Y_0$.
- At the end, third frame is rotated by an angle α around $-Z_0$.

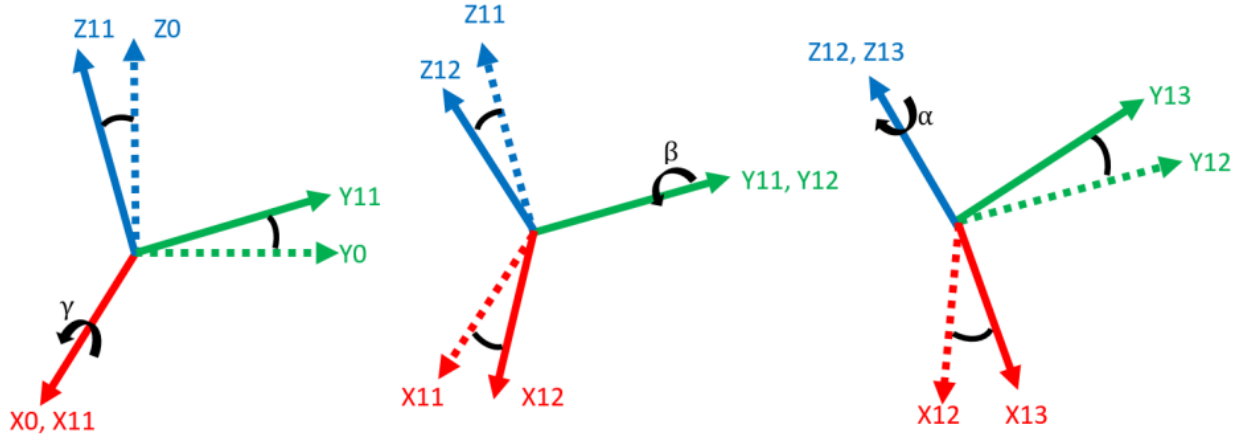


Fig. 3.3. Euler convention rotations $R_X(\gamma)$, $R_Y(\beta)$ and $R_Z(\alpha)$

A more universal and understandable depiction of the end-effector orientation is made possible by the Euler angular convention. The rotations of Euler towards the (X, Y, Z) base frame are (α, β, γ), respectively. They can be calculated using the terms from the homogeneous transformation matrix described in (1), which are (3) through (5).

$$\beta = \arctan_2(-r_{13}, \sqrt{r_{11}^2 + r_{21}^2}) \quad (3)$$

$$\alpha = \arctan_2\left(\frac{1}{r_{21}} \cos(\beta), \frac{1}{r_{11}} \cos(\beta)\right) \quad (4)$$

$$\gamma = \arctan_2\left(\frac{1}{r_{32}} \cos(\beta), \frac{1}{r_{33}} \cos(\beta)\right) \quad (5)$$

where the homogeneous transformation matrix of the manipulator described in (1) contains the r_{ij} terms. As in (6), the goal of the orientation's single-objective optimisation is to reduce the generated solution's Euclidean distance from the desired orientation.

$$\min f_2(Q) = \frac{|\gamma - \gamma_t| + |\alpha - \alpha_t| + |\beta - \beta_t|}{3} \quad (6)$$

3.4. Inverse Kinematics (IK):

In simple terms, inverse kinematics is forward kinematics in reverse. If the end effector's position and orientation are known, IK fundamentally refers to the process of finding the joint parameters of the manipulator. For certain robotics activities, such as moving items along a specific path or performing picking and placing operations, inverse kinematics is required. Due to the fact that the end effector location is not always the main concern, inverse kinematics can be considered as a multi-objective problem. As opposed to forward kinematics, the inverse kinematics solution can lead to multiple configurations for the same end effector posture, each of which corresponds to a different joint position vector. As a result, inverse kinematics is a more challenging problem than forward kinematics. There are several approaches for solving inverse kinematics problem:

- Analytical Approach
- Numerical Approach

For our 5D0F Robot, numerical solution is incredibly difficult to find as the non-linearities in the equation increases drastically and a viable mathematical model is tough to equate. Therefore we will be using Analytical Approach for our application. The inverse kinematics model considering orientation may be formulated as in (7) assuming that the joints coordinate is $Q()$, which may be rotations or translations as well as angular or linear velocities; that the target position is X_t ; and that the forward kinematics model of the system is in (1).

To find the function satisfying both the functions given in (7)

$$\min f_1(Q) = \min \sqrt{(p_x - x_t)^2 + (p_y - y_t)^2 + (p_z - z_t)^2}$$

$$\min f_2(Q) = \frac{|\gamma - \gamma_t| + |\alpha - \alpha_t| + |\beta - \beta_t|}{3} \quad \text{Subject to : } Q \in Q_{DS} \quad (7)$$

In which $f_1(Q)$ is the target location's Euclidean distance from the location of the end-effector, which is subject to minimization. The target orientation to the end-effector orientation's Euclidean distance, $f_2(Q)$, is likewise subject to reduction. The destination position's coordinates are (x_t, y_t, z_t) . The gripper position for a particular joint position (Q) is $P = (p_x, p_y, p_z)$. The desired orientation of the end-effector is represented by the target orientation (t, t, t) . The first aim, which is connected to f_1 , tries to reduce the distance between the position of the target and the gripper. The second aim, which corresponds to f_2 , aims to satisfy specific effector orientation constraints. In order to get the most accurate orientation to the desired orientation, orientation optimization tries to reduce the mean of all angular errors.

Chapter 4

Simulation

To construct the model in Simulink in order to implement forward kinematics block (get Transform block in Robotics Toolbox) we need to make a Rigid Body Tree (RBT). As the name suggests a RBT is made up of rigid bodies connected via joints.

“Fig. 4” shows the RBT of the robot. In order to reduce complexity, we have only modeled from the Base to Gripper Base, this is because the gripper contains cylindrical joints whose RBT is hard to evaluate.

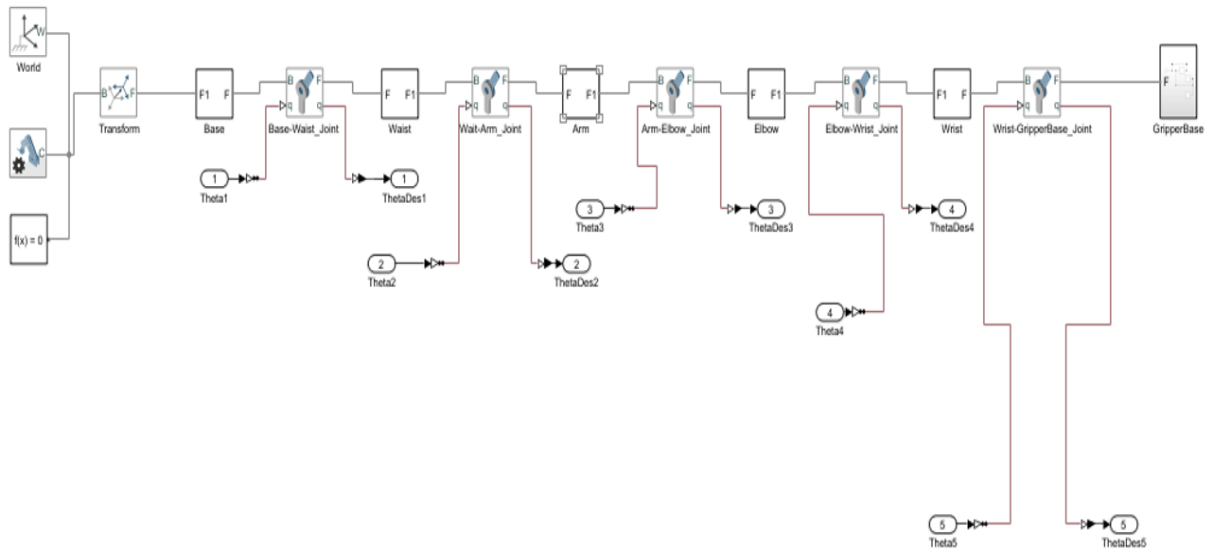


Fig. 4.1. Rigid Body Tree of the Manipulator

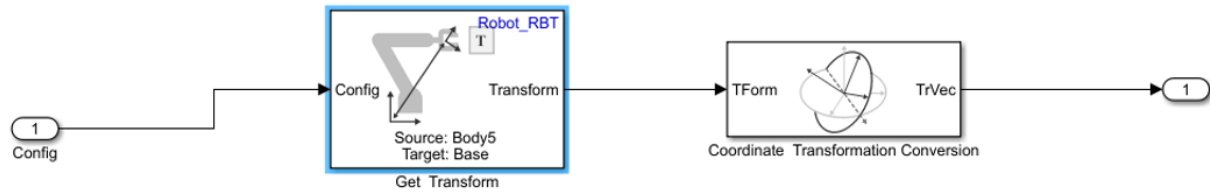


Fig. 4.2 Forward Kinematics block

Sampling time $T_s = 0.001$ sec

Total Simulation Time: 8.67 sec

RBT is stored in a variable named Robot_RBT

Configure input accepts a 5x1 vector of all joint angles

Theta1-Theta5 are joint angles calculated by forward kinematics block

ThetaDes1-ThetaDes5 are joint angles after driving through RBT or Physical model of the Robot with inertial considerations

The output of the Get Transform block is in the form of a Homogeneous Transformation Matrix of 4x4, which we need to convert to Translation Vector $[x \ y \ z]$ in order to plot the desired results.

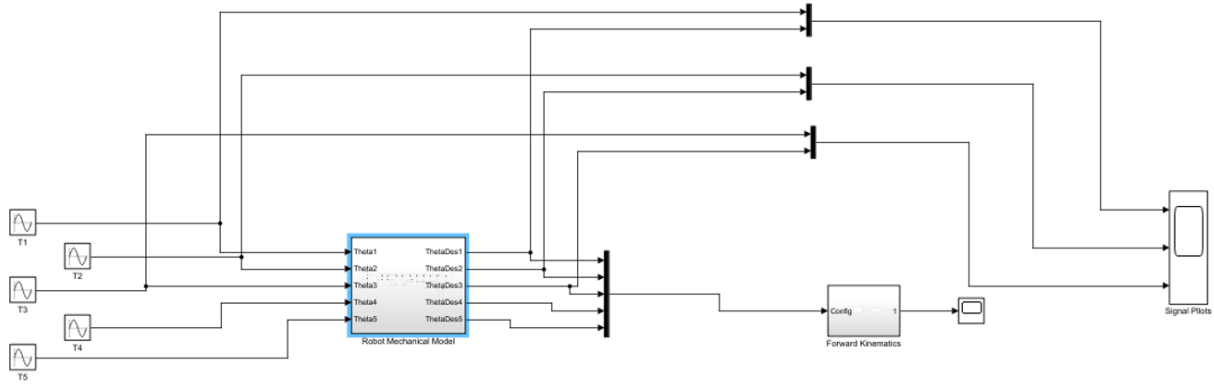


Fig. 4.3. The complete Forward Kinematic Implementation

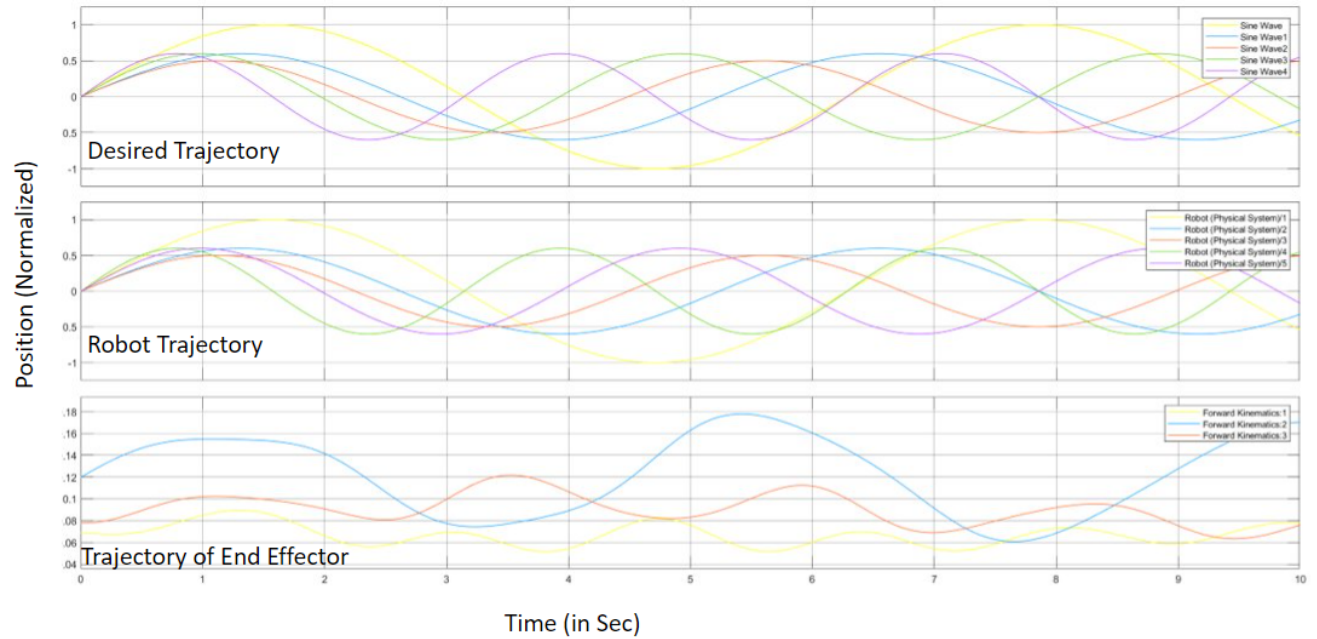


Fig 4.4. Plot1: Y Axis – Amplitude; X-axis – Time in seconds

Since the forward kinematics block is an open loop system and inertia is set to zero, $\theta = \theta_{Des}$ for all values of joint variables.

Chapter 5

Trajectory Planning

Motion planning is a crucial term in robotics that involves decomposing a desired movement task into smaller motions that adhere to movement constraints and can potentially optimize some aspect of the movement. It is one of the three key components that enable systems like autonomous cars, robot manipulators, UAVs, and UGVs to function independently. The other two components are perception and control..

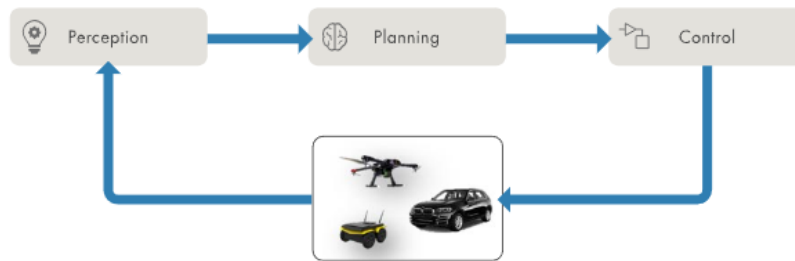


Fig 5.1. Autonomous System Workflow

Similar to humans, autonomous systems initially scan their surroundings to obtain information about their location and the objects in their environment. Once they have a map of the environment, motion planning algorithms create a path that is clear of obstacles and leads to a specified destination. The system's actuators are then controlled by a controller, which sends commands based on the next step decision made on this path

5.1. What Is Motion Planning?

Motion planning is a computational problem to find a sequence of actions that moves a robot or vehicle from an initial state to a goal state. “Motion planning” and “path planning” are often used interchangeably, but there is a key difference. Motion planning generates the vehicle’s motion as it changes position over time, whereas path planning only generates a path for the vehicle.

5.2. Common Types of Motion Planning

There are many different types of approaches to motion planning. The most common are:

Search-based planning and sampling-based planning approaches, which are based on the way the search tree or graph is created.

Global and local path planning approaches, which are based on whether the planning is done in the entire map or in a subset.

Trajectory planning for robots involves determining the path and motion of a robot to achieve a desired goal or task. This involves breaking down the task into smaller sub-tasks and determining the sequence of movements required to complete each sub-task.

The trajectory planning process involves several steps:

Task Specification: Define the desired goal or task that the robot needs to achieve. This could involve specifying the position, orientation, and other constraints that the robot needs to follow.

Kinematic Analysis: Determine the configuration of the robot and the constraints on its motion. This includes identifying the degrees of freedom (DOFs) of the robot, the range of motion for each DOF, and the joint velocities and accelerations that the robot can achieve.

Path Planning: Determine the path that the robot needs to follow to achieve the desired task. This could involve using algorithms such as A* search, RRT (Rapidly-exploring Random Trees), or potential field planning.

Trajectory Generation: Generate a smooth trajectory that the robot can follow to achieve the desired path. This involves computing the joint positions, velocities, and accelerations required to follow the path.

Collision Avoidance: Ensure that the generated trajectory does not result in any collisions between the robot and other objects in the environment. This can be achieved by using sensors such as cameras, LIDAR, or ultrasonic sensors to detect obstacles and adjust the trajectory accordingly.

Execution: Once the trajectory has been generated, the robot can execute the task by following the planned path and trajectory. This could involve controlling the robot using a variety of techniques such as PID control, model predictive control, or machine learning algorithms. The problem of navigation or motion planning as a whole includes trajectory planning as a subproblem. Following is a typical hierarchy for motion planning:

1. **Task planning** - It is the process of creating a list of high-level targets, for example "pick up the object in front of you."

2. **Path planning** - It is the process of creating an appropriate path from one point to another. The waypoints along a path are frequently connected to one another.
3. **Trajectory planning** – It is the process of creating a proper schedule in what manner a path will be followed along with position, velocity, and acceleration restrictions.

5.3. Types Of Trajectories:

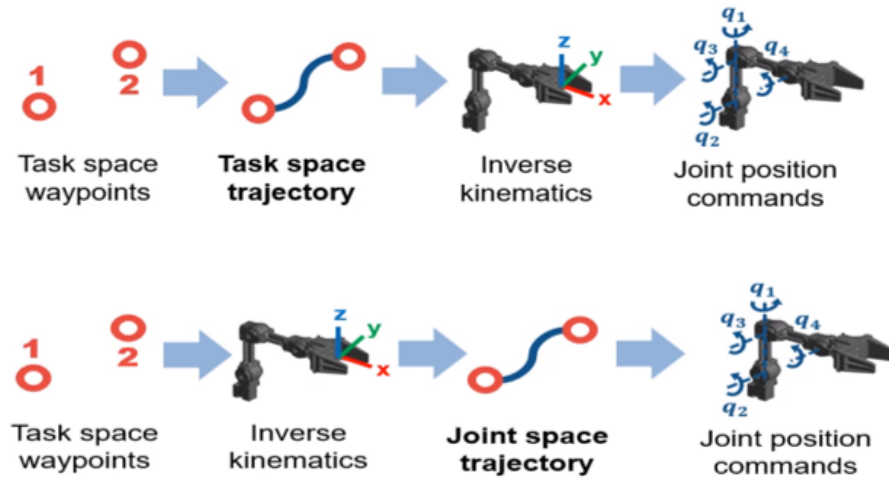


Fig 5.2. Task Space vs Joint Space Trajectory

- Task space denotes that the waypoints and interpolation are based on a particular Cartesian pose (position and orientation) on the robotic arm, typically the end effector.
- Joint space denotes that the interpolation and waypoints are based directly on the positions of the joints or angles depending on the kind of joint.

The primary distinction between task space and joint space trajectory is that the task-space trajectories resemble more "natural" motions than joint-space trajectories do. This is due to the smooth movements of the end effector in relation to the surroundings in spite of not so smooth movement of the joints. The main disadvantage of a task-space trajectory is that in it, inverse kinematics is solved more frequently compared to a joint-space trajectory. This necessitates a

significant amount of computation, particularly if the solver for inverse kinematics is focused on optimisation.

Table 2. Joint Space Vs Task Space Trajectories

	Task Space	Joint Space
Advantages	<ul style="list-style-type: none"> • Motion is anticipated (in task space interpolation). • Improved collision and obstacle avoidance. 	<ul style="list-style-type: none"> • Inverse Kinematics is calculated just at waypoints so it takes less execution time. • Movement of the actuator is fluent and simpler to verify.
Disadvantages	<ul style="list-style-type: none"> • Inverse Kinematics is calculated just at every time step so it takes more time for execution. • Movement of the actuator is less predictable and more challenging to verify. 	<ul style="list-style-type: none"> • There is no guarantee that intermediate points will adhere to joint restrictions or be free from collisions.

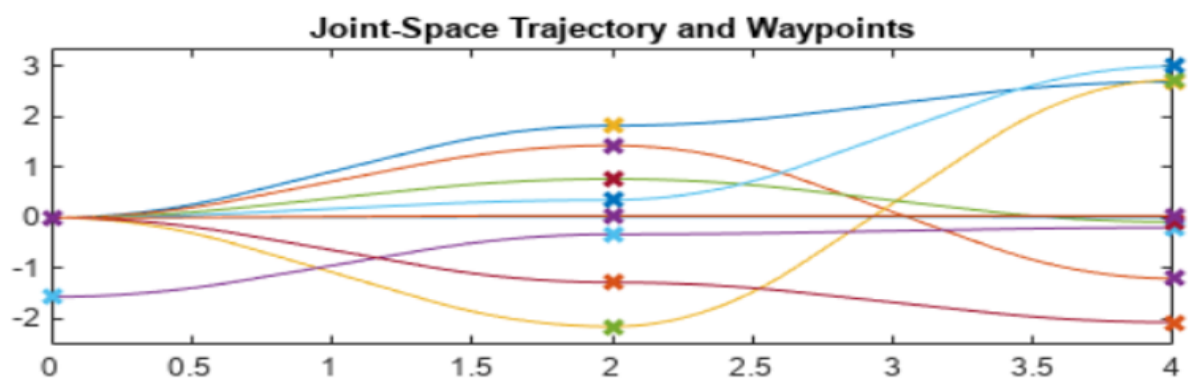


Fig 5.3 Joint Space Trajectory and waypoints

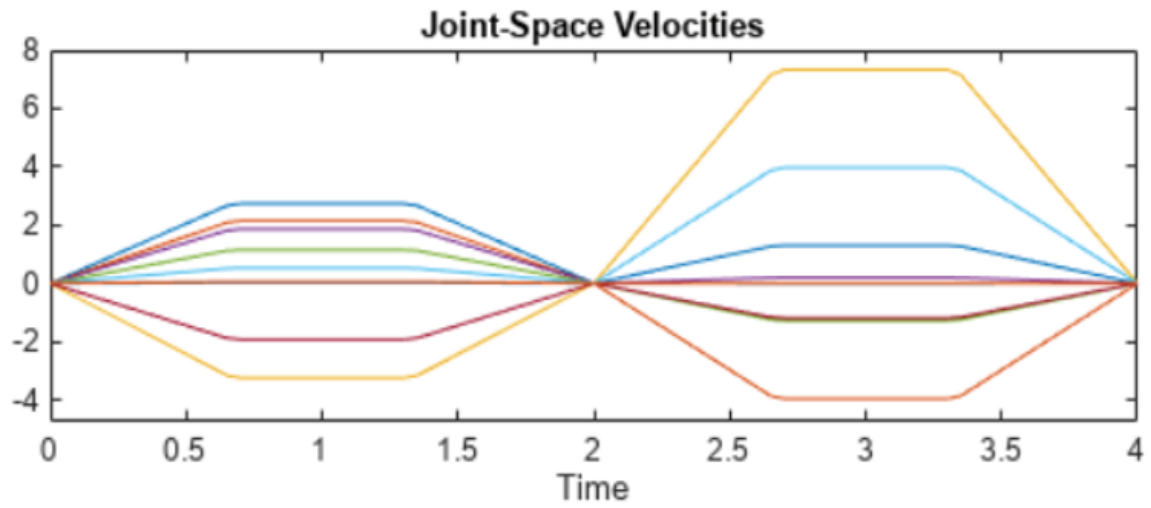


Fig 5.4 Joint-Space Velocities

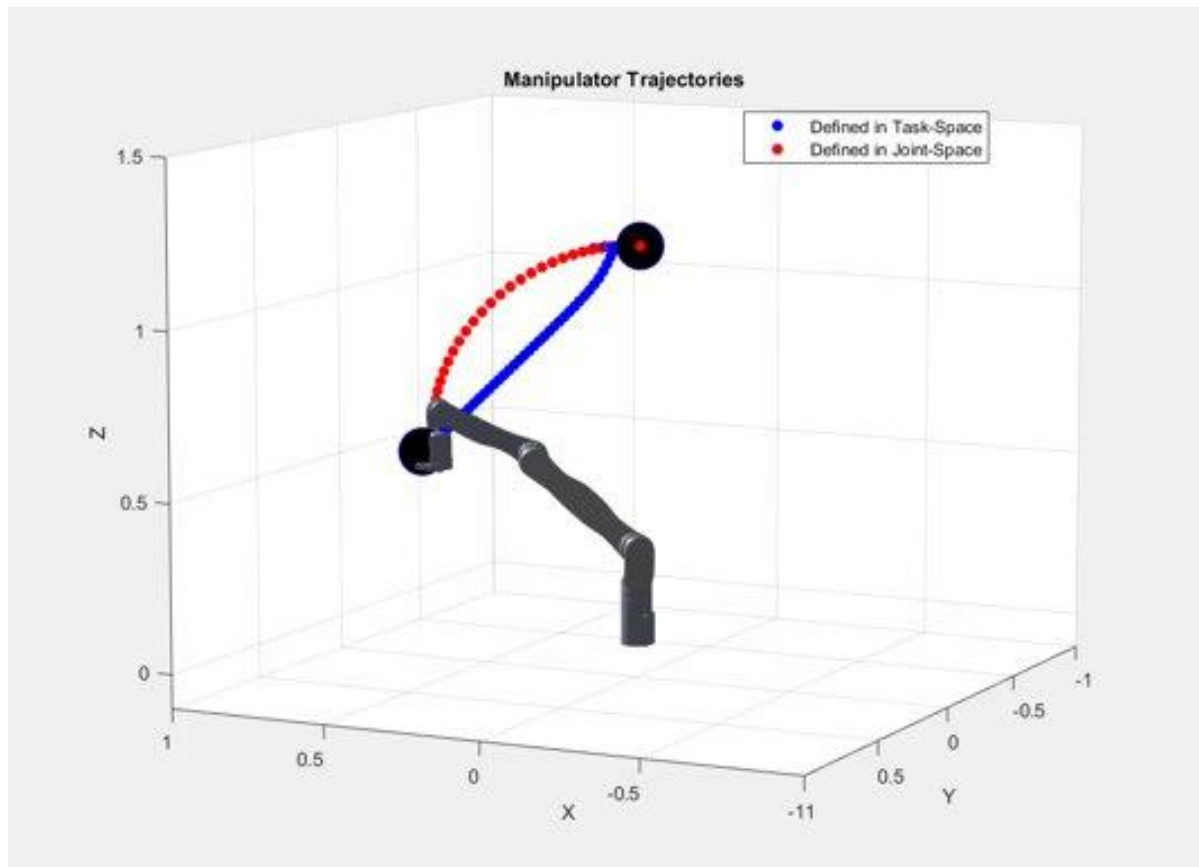


Fig 5.5. Task Space and Joint Space Simulation

5.4. Trajectory generation techniques:

5.4.1 Trapezoidal trajectory:

A smooth point-to-point motion is guaranteed by a trapezoidal velocity profile, which pauses at every waypoint. The name of the profile is based on 3 stages of each segment, which links two waypoints. Those three stages are as follows :

- Acceleration ranging from rest to maximum velocity
- Maximum velocity with constant speed
- Deceleration to zero velocity

As a result, the velocity distribution over every segment is shaped like a trapezium. The end time, peak velocity, peak acceleration, and acceleration time characteristics each identify a segment, although only two of these elements are necessary to completely describe the motion.

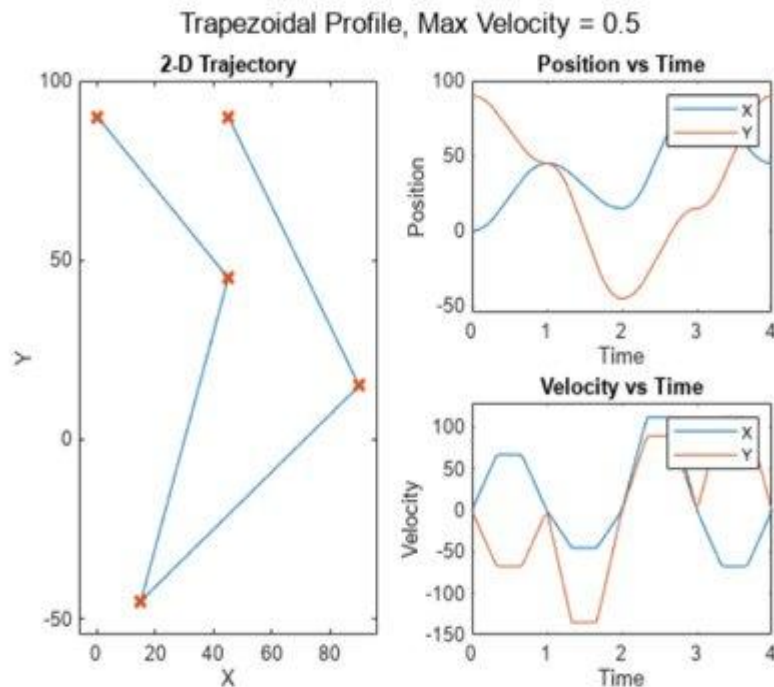


Fig. 5.6. Trapezoidal Profile of Manipulator Arm

5.4.2 Polynomial trajectory:

Polynomials of different orders can be used to interpolate between two waypoints. In practise, the following orders are most frequently used:

- **Cubic** - It is a polynomial of 3rd degree in which four boundary constraints are required including position and velocity at both ends.
- **Quintic** - It is a polynomial of 5th degree in which six boundary constraints are required including position, velocity and acceleration at both ends.

A cubic polynomial trajectory is a type of trajectory that can be used to describe the motion of a robot or other object. It is a mathematical function that can be used to generate a smooth path or trajectory by defining the position, velocity, and acceleration of the object at each point in time.

The basic form of a cubic polynomial trajectory can be written as follows:

$$p(t) = A + Bt + Ct^2 + Dt^3$$

Where $p(t)$ is the position of the object at time t , A represents the starting location of the object, B represents the initial velocity of the object, C represents the acceleration of the object, and D is a constant that determines the curvature of the trajectory.

To generate a cubic polynomial trajectory, we typically need to specify the initial and final positions of the object, as well as the initial and final velocities. We can then solve for the values of C and D using these constraints, resulting in a unique trajectory that satisfies the given constraints.

One advantage of cubic polynomial trajectories is that they can be easily differentiated to obtain the velocity and acceleration of the object at any point in time. This makes them useful for controlling the motion of robots or other objects in real-time, as we can adjust the trajectory based on feedback from sensors or other sources. Additionally, cubic polynomial trajectories are computationally efficient to compute and implement, making them a popular choice for many robotics applications.

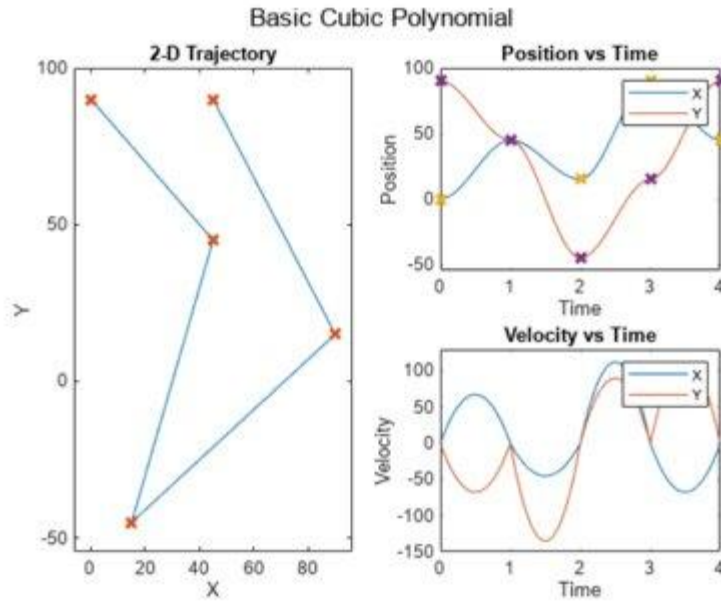


Fig. 5.7 Basic cubic polynomial Trajectory Profile of Manipulator arm

5.5 Rotation interpolation:

Up until now, we have only demonstrated how to create trajectories in position, but controlling how the end effector is oriented is also important. However, interpolating orientation can be more complicated than interpolating position because angles continuously wrap and few presentations of orientations, for example the case of Euler angles, in which the same configuration can be represented in several ways. One solution to this problem is to use quaternions to interpolate orientation, which provide an unambiguous way to represent orientation. The popular method for interpolating orientations using quaternions is known as Spherical Linear Interpolation (Slerp), which determines the shortest route between two orientations while maintaining a constant angular velocity about a fixed axis.

Although Slerp takes into account linear interpolation with a constant velocity, we can modify the behavior of the trajectory by incorporating time scaling. Instead of uniformly spacing out the time vector, you can use the trajectory warping techniques discussed earlier to alter it. For instance, applying a trajectory with a trapezoidal velocity time scaling will have zero velocity at the beginning and end of each segment and maximum velocity in the center.

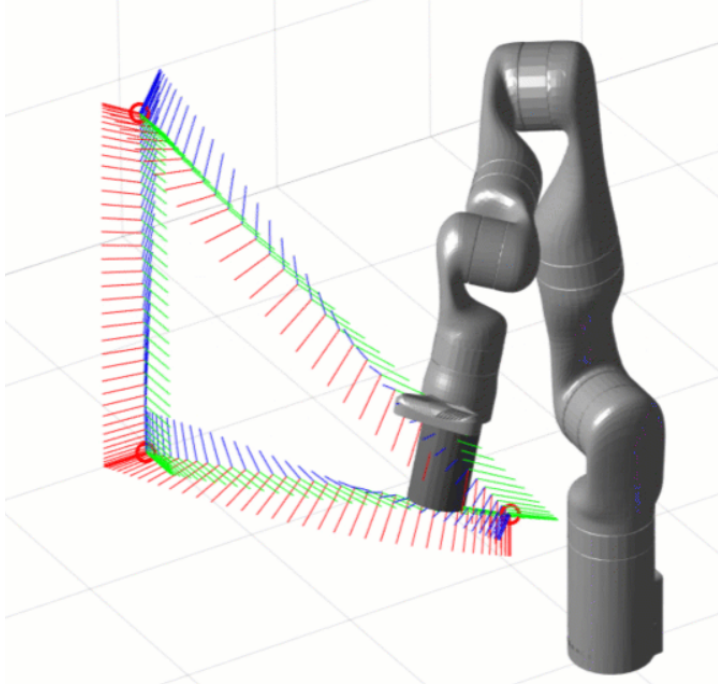


Fig 5.8. Rotating End Effector using SLERP Method (Kenova Gen 3)

Chapter 6

Perception

Perception is a critical component of robotics vision that involves processing sensor data to extract useful information about the environment. In robotics, perception systems are typically used to detect and locate objects, estimate distances, and track movements.

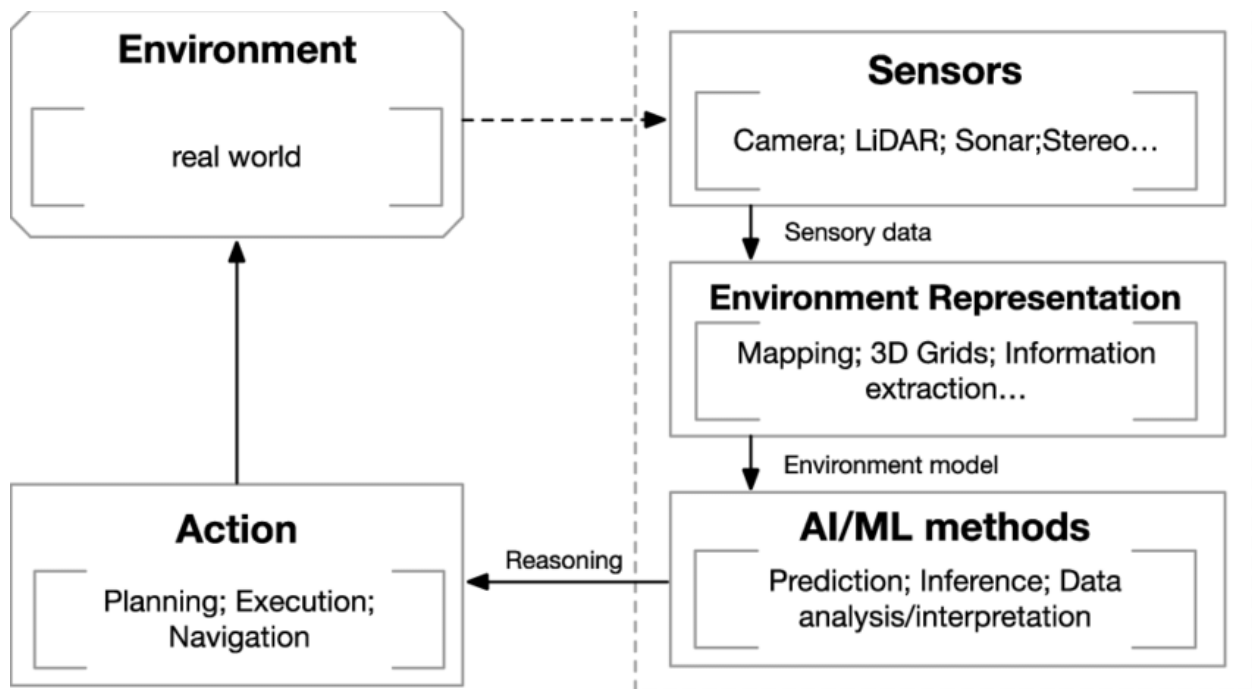


Fig. 6.1. Perception for intelligent robot

Vision is one of the most common sensor modalities used for perception in robotics. Vision-based perception involves using cameras and image processing techniques to extract information from visual data. This can be achieved using a variety of algorithms, including object detection and tracking, image segmentation, and feature extraction.

Object detection and tracking is a common technique used in robotics vision for identifying and tracking objects in the environment. This involves using machine learning algorithms to identify specific objects in images or video feeds and tracking their movements over time. Object detection and tracking can be used for a variety of uses, such as self-driven navigation, object handling, and human-robot communication etc.

Image segmentation is another important technique used in robotics vision for identifying and separating different objects in an image or video feed. This involves dividing the image into distinct regions based on color, texture, or other features, and then grouping those regions into objects. Image segmentation is useful for applications such as object recognition, scene understanding, and depth estimation.

Feature extraction is used in robotics vision for identifying and extracting relevant features from an image or video feed. This involves identifying key points or patterns in the image that can be used to identify or track objects. Feature extraction can be used for a variety of applications, including object recognition, pose estimation, and scene reconstruction.

Perception is essential for enabling robots to interact effectively with their environment. By accurately perceiving the environment, robots can make informed decisions about how to move and interact with objects in the environment. Additionally, perception can help robots avoid collisions with obstacles and other objects, ensuring safe and effective operation.

In this Project we are going to use an android phone as a webcam, live streaming its video onto the web server as an IP address. Using a single camera we will estimate the distance of the target from the origin of the image

6.1. Coordinate Estimation of Target T

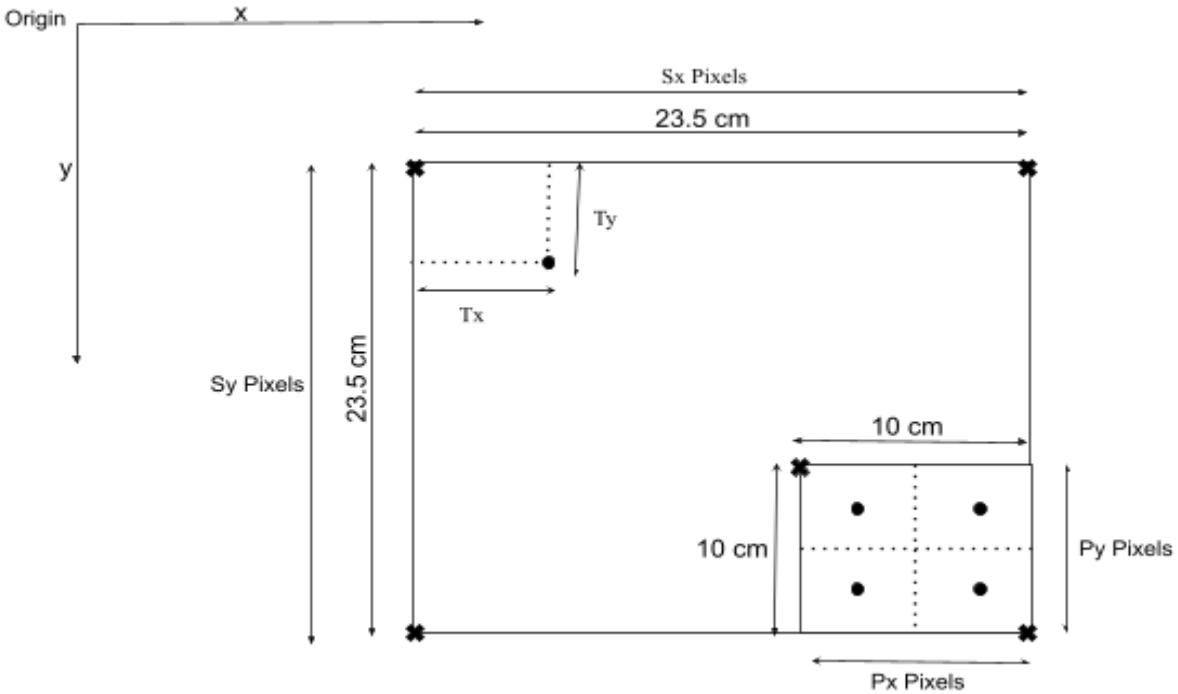


Fig. 6.2. Platform for detecting cubes (S_x, S_y), Placement Station for stacking/sorting the cubes (P_x, P_y)

To find the points of the rectangle :-

$$P_y \text{ Pixels} = 10 \text{ cm}$$

$$P_x \text{ Pixels} = 10 \text{ cm}$$

$$S_y \text{ Pixels} = 23.5 \text{ cm}$$

$$S_x \text{ Pixels} = 23.5 \text{ cm}$$

$$T_y \text{ Pixels} = 22 \text{ cm}$$

$$T_x \text{ Pixels} = \frac{23.5}{S_y} * T_y$$

For better estimation :

$$T_x \text{ Pixels} = \left(\frac{23.5}{S_x} + \frac{10}{P_x} \right) * \frac{T_x}{2}$$

$$T_y \text{ Pixels} = \left(\frac{23.5}{S_y} + \frac{10}{P_y} \right) * \frac{T_y}{2}$$

6.2. Perception System Design

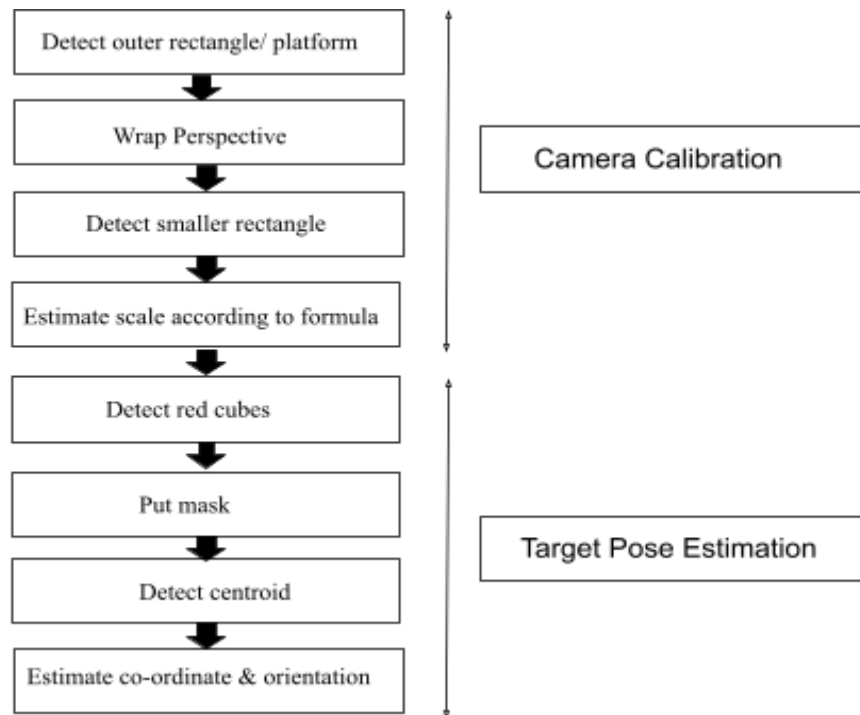


Fig. 6.3. Perception Workflow

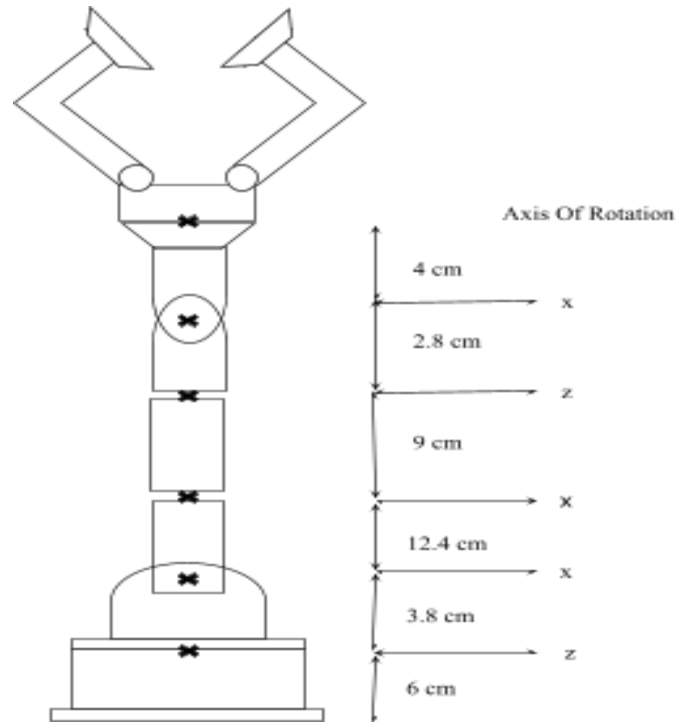


Fig. 6.4. A simplified model of robotic arm for intuitive estimation of the robot position

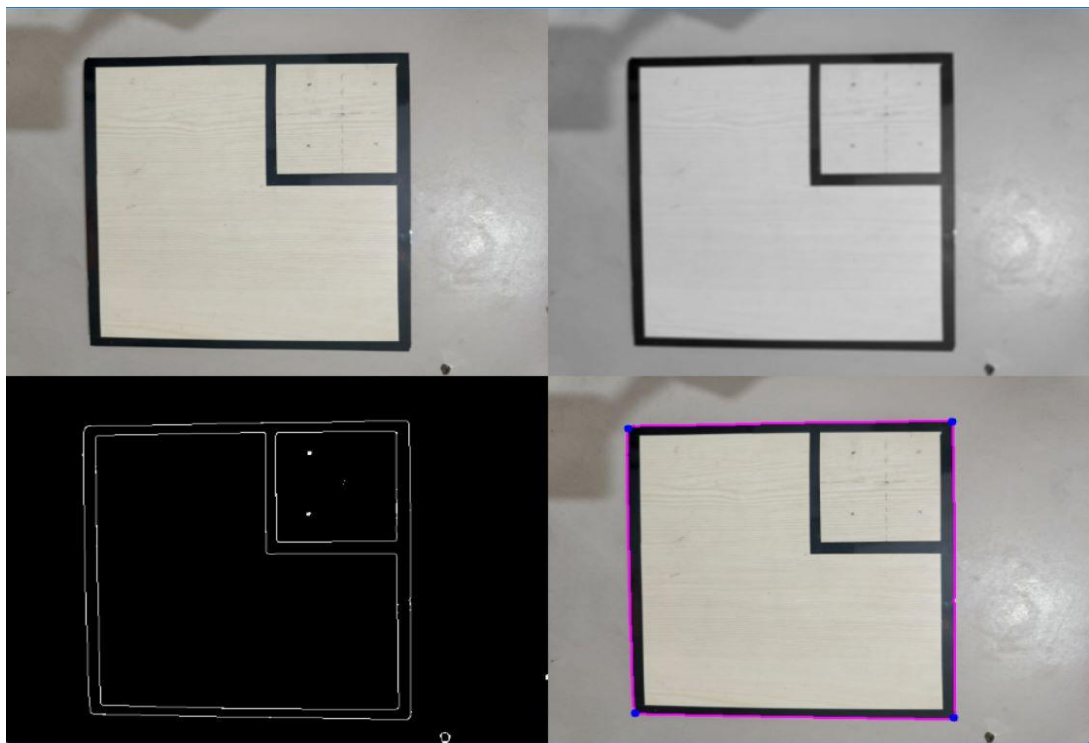


Fig. 6.5. Placement Station detection (a) Original Image (b) Gaussian Blur (c) Canny Edge Detector (d) Contour Detection [Image size: 620x480 pixels]

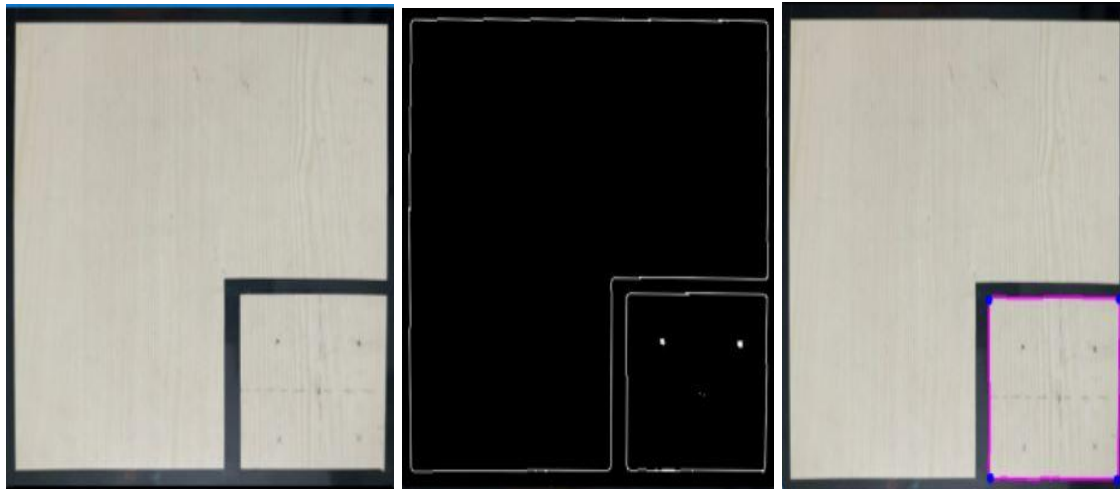


Fig.6.6. Placement Station (a) Cropped Platform Image (Distortion removed) (b) Canny edge detection (c) Placement Station Detection [Image size: 500x500 pixels]

Blue dots represent the coordinate of the corner of placement station

6.3. Estimate Coordinate of Target T:

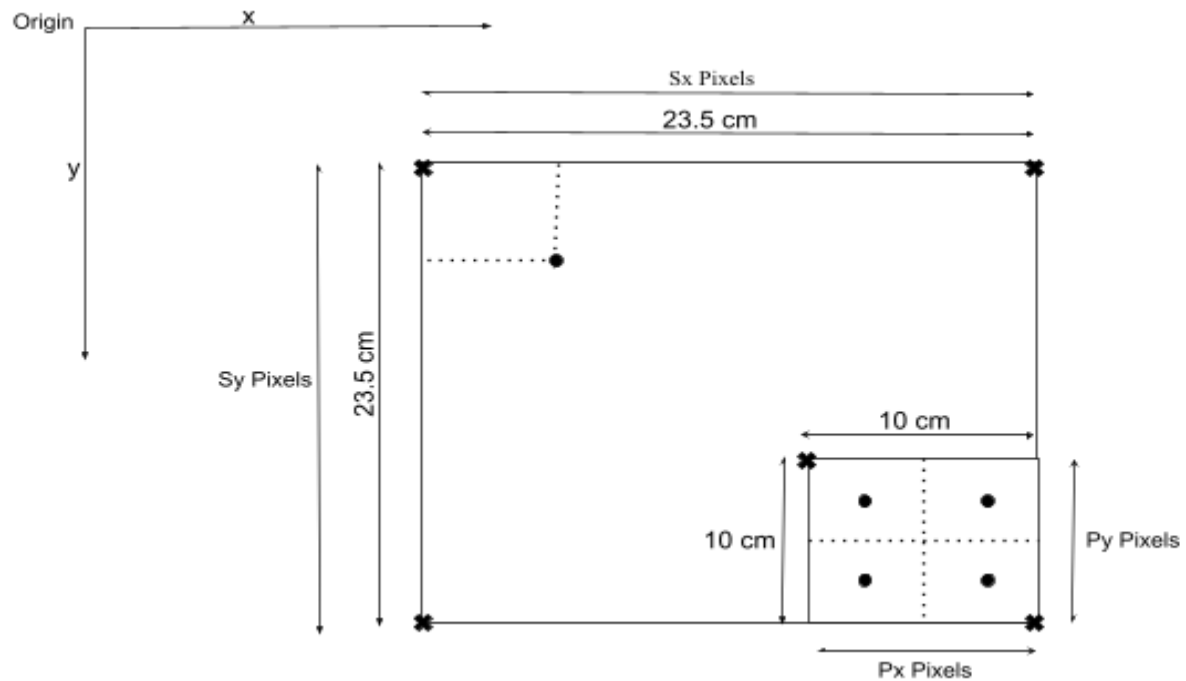


Fig.6.7. Platform for detecting cubes (S_x, S_y), Placement Station for stacking/sorting the cubes (P_x, P_y)

To find the points of the rectangle :-

$$P_y \text{ Pixels} = 10 \text{ cm}$$

$$P_x \text{ Pixels} = 10 \text{ cm}$$

$$S_y \text{ Pixels} = 23.5 \text{ cm}$$

$$S_x \text{ Pixels} = 23.5 \text{ cm}$$

$$T_y \text{ Pixels} = 22 \text{ cm}$$

$$T_x \text{ Pixels} = \frac{23.5}{S_y} * T_y$$

For better estimation :

$$T_x \text{ Pixels} = \left(\frac{23.5}{S_x} + \frac{10}{P_x} \right) * \frac{T_x}{2}$$

$$T_y \text{ Pixels} = \left(\frac{23.5}{S_y} + \frac{10}{P_y} \right) * \frac{T_y}{2}$$

Average pixel distance between each corner of the station: 183.7 pixels

Estimated scale : 9cm/183.7pixels = 0.049cm/pixel

Calculated scale : 25cm/500pixels = 0.05cm/pixel

Estimation error : 2.01%

6.4. Cube Detection:



Fig 6.8. Colour Masking (a) Original Image (b) Mask (c) Result (hsv value for red cube h=0 s=108 v=4)

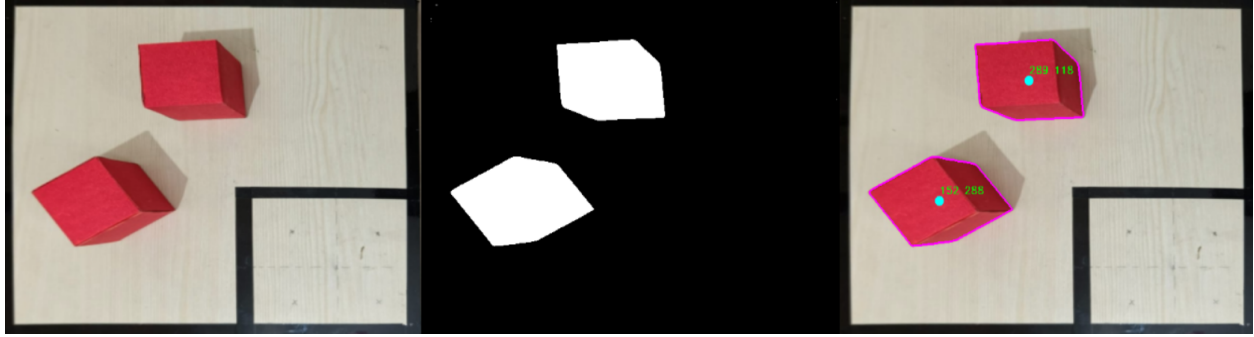


Fig 6.9. Cropped Platform: (a) Original Image (b) Mask (c) Result Co-ordinates

Chapter 7

Hardware Implementation

The Work Flow of the Pick & Place Stack operation is given below, under Path planning, we are going to pick the cube closest to the current configuration first , and this min distance strategy is applied to other cubes till all the cubes are inside the placement station. The servo motors are driven using arduino uno, with the help of python library pyfirmata, respecting a trapezoidal velocity profile in joint-space. A 5v 2amp DC power supply is used to drive current to the motors.

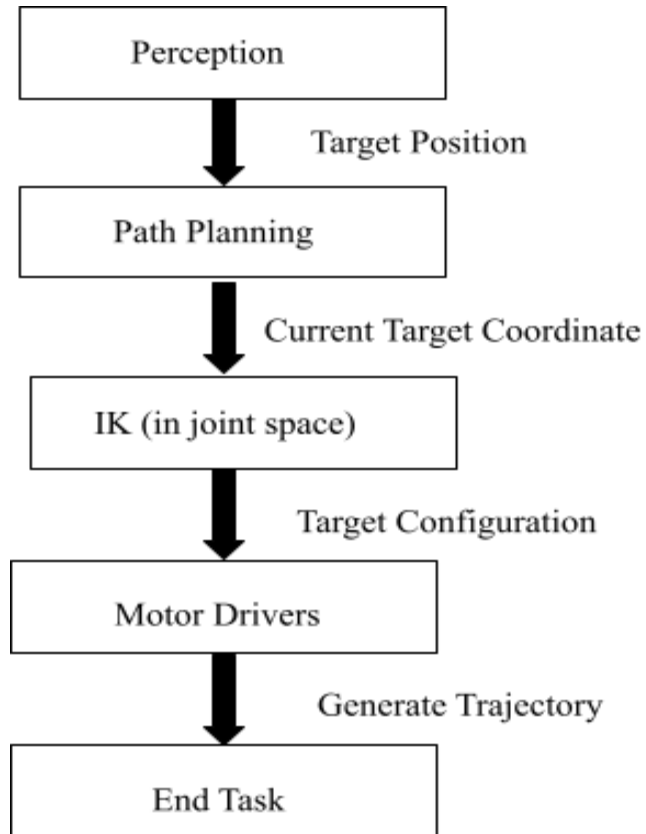
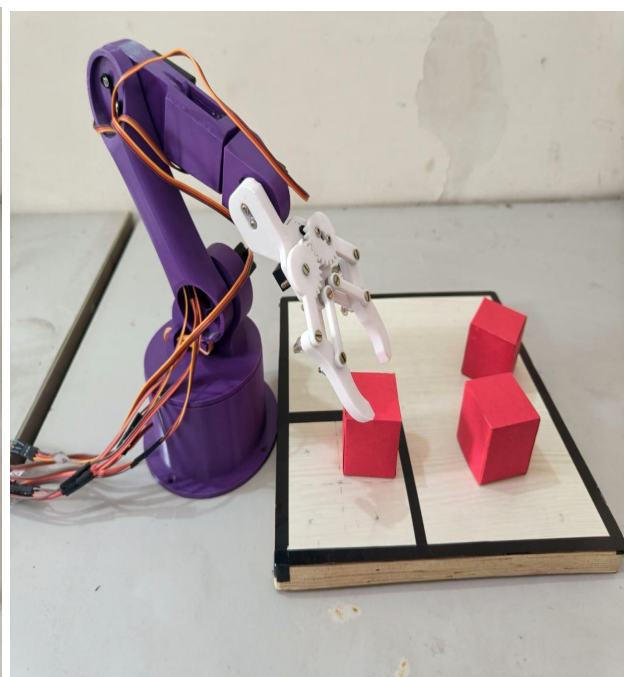
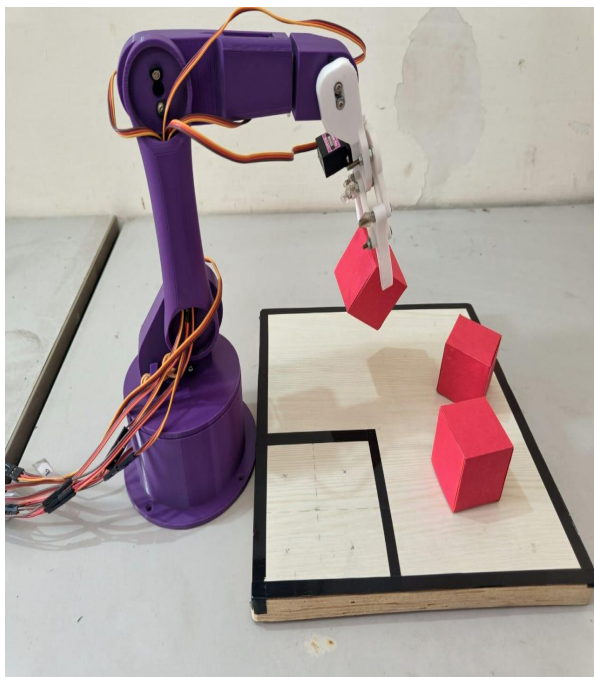
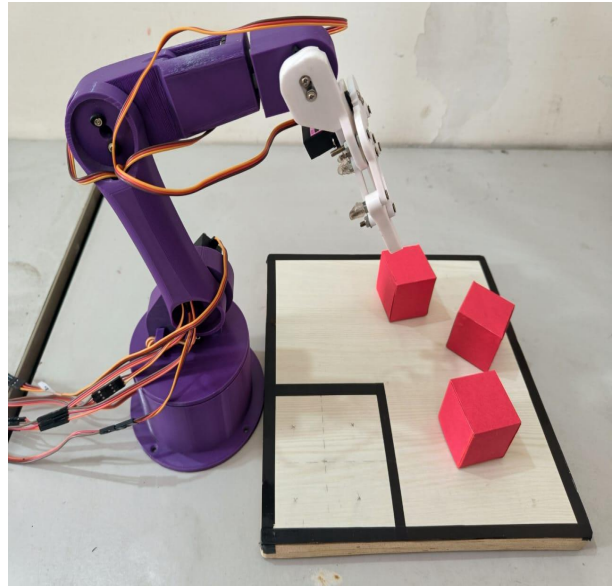


Fig. 7.1 Work Flow of pick place and stack operation

7.1. Process Of palletizing:



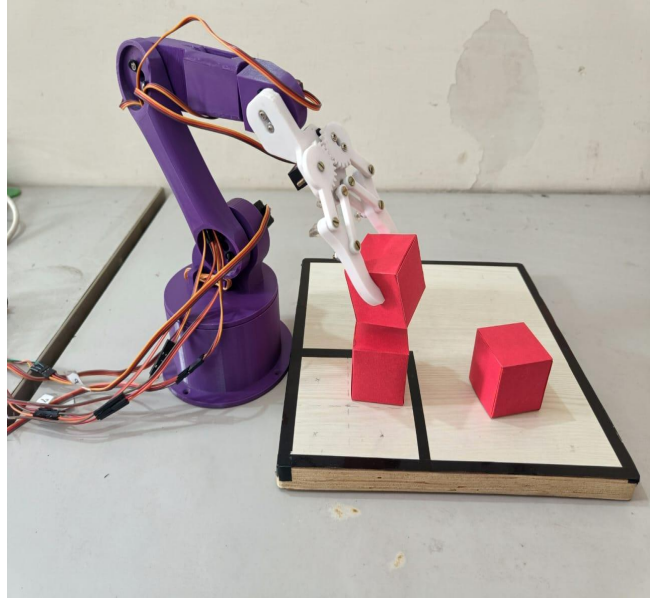
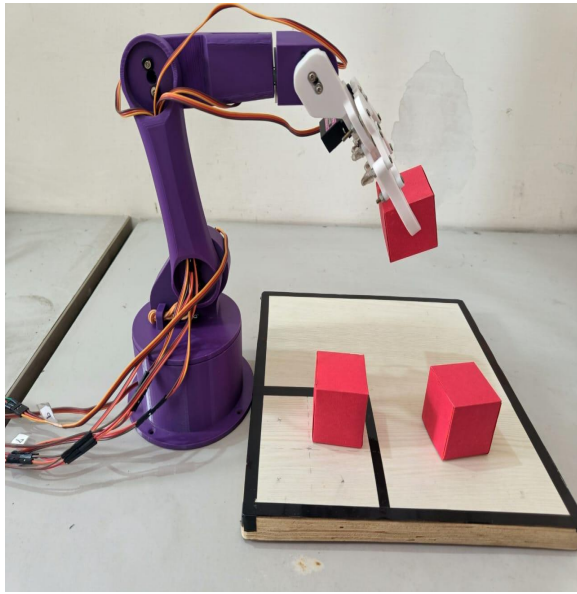


Fig. 7.2. 5 DOF Robotic Arm performing task of palletization

Chapter 8

8.1. Conclusion:

We have examined a variety of trajectory generation algorithms. We have chosen the Trapezoidal Velocity Profile due to the ease of interpolation and the absence of a problem with exceeding velocity limits. As a result of our literature review and implementation, we have determined that numerical solutions of Inverse Kinematics are one of the most difficult and time-consuming problems in robotics; therefore, we will use joint-space trajectory to reduce the number of points that must be solved inversely. In addition, since no collision objects are considered for our original object palletization task for the sake of project simplicity, our path planning is limited to scheduling tasks. The hobby-use servo motors used in this project have an integrated position controller, and torque control is challenging to implement, so a separate position controller is unnecessary. The perception system that can detect the coordinates and position of an object to be lifted up is mostly based on computer vision techniques such as canny edge detection, hough line transforms, warp perspectives and noise filtering using gaussian blur.

8.2. Scope of research:

Considering the simplicity of the hardware of the perception system, there is potential to use Lidar based sensing for more accurate distance estimation. Moreover if the IK function can be more optimized to work within the platform bounds only, thereby reducing further complexity. Collision avoidance can be introduced to reduce the risk of objects colliding with the robot which are not the target, thereby disturbing the placement location and positions. Orientation

detection and detection of best grasping location for different types of objects such as cuboids, cylinders etc can be introduced in order to generalize the organizing task.

References

- [1]. Jiang, D., Li, G., Sun, Y. et al. Manipulator grabbing position detection with information fusion of color image and depth image using deep learning. *J Ambient Intell Human Comput* 12, 10809–10822 (2021). <https://doi.org/10.1007/s12652-020-0284>
- [2]. Ahmadi, S. (2022). Real-time motion planning of 6 DOF Collaborative Robot (Dissertation). Retrieved from <http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-308528>
- [3]. Antti Hietanen, Jyrki Latokartano, Alessandro Foi, Roel Pieters, Ville Kyrki, Minna Lanz, Joni-Kristian Kämäräinen, "Benchmarking pose estimation for robot manipulation", *Robotics and Autonomous Systems*, Volume 143, 2021, 103810, ISSN 0921-8890, <https://doi.org/10.1016/j.robot.2021.103810>.
- [4]. Rafal Szczepanski, Krystian Erwinski, Mateusz Tejer, Artur Bereit, Tomasz Tarczewski, "Optimal scheduling for palletizing task using robotic arm", *Engineering Applications of Artificial Intelligence*, Volume 113, 2022, 104976, ISSN 0952-1976, <https://doi.org/10.1016/j.engappai.2022.104976>.
- [5]. Rokbani N, Neji B, Slim M, Mirjalili S, Ghandour R. A Multi-Objective Modified PSO for Inverse Kinematics of a 5-DOF Robotic Arm. *Applied Sciences*. 2022; 12(14):7091. <https://doi.org/10.3390/app12147091>
- [6]. Gi Hyun Lim, Nuno Lau, Eurico Pedrosa, Filipe Amaral, Artur Pereira, José Luís Azevedo & Bernardo Cunha (2019) Precise and efficient pose estimation of stacked objects for mobile manipulation in industrial robotics challenges, *Advanced Robotics*, 33:13, 636-646, DOI: <https://doi.org/10.1080/01691864.2019.161778>
- [7]. Nadeau, (2019). Pybotics: Python Toolbox for Robotics. *Journal of Open Source Software*, 4(41), 1738, <https://doi.org/10.21105/joss.01738>
- [8]. Liangjun Zhang, Y. J. Kim and Dinesh Manocha, "A hybrid approach for complete motion planning," 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007, pp. 7-14, DOI: 10.1109/IROS.2007.4399064.
- [9]. Gi Hyun Lim, Nuno Lau, Eurico Pedrosa, Filipe Amaral, Artur Pereira, José Luís Azevedo & Bernardo Cunha (2019) Precise and efficient pose estimation of stacked objects for mobile manipulation in industrial robotics challenges, *Advanced Robotics*, 33:13, 636-646, DOI: 10.1080/01691864.2019.1617780
- [10]. Jiang, D., Li, G., Sun, Y. et al. Manipulator grabbing position detection with information fusion of color image and depth image using deep learning. *J Ambient Intell Human Comput* 12, 10809–10822 (2021). <https://doi.org/10.1007/s12652-020-02843-w>
- [11]. Ahmadi, S. (2022). Real-time motion planning of 6 DOF Collaborative Robot (Dissertation). Retrieved from <http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-308528>

- [12]. Antti Hietanen, Jyrki Latokartano, Alessandro Foi, Roel Pieters, Ville Kyrki, Minna Lanz, Joni-Kristian Kämäräinen, "Benchmarking pose estimation for robot manipulation", *Robotics and Autonomous Systems*, Volume 143, 2021, 103810, ISSN 0921-8890, <https://doi.org/10.1016/j.robot.2021.103810>.
- [13]. Rafal Szczepanski, Krystian Erwinski, Mateusz Tejer, Artur Bereit, Tomasz Tarczewski, "Optimal scheduling for palletizing task using robotic arm", *Engineering Applications of Artificial Intelligence*, Volume 113, 2022, 104976, ISSN 0952-1976, <https://doi.org/10.1016/j.engappai.2022.104976>.
- [14]. Reinhard Klette (2014). *Concise Computer Vision*. Springer. ISBN 978-1-4471-6320-6.
- [15]. Linda G. Shapiro; George C. Stockman (2001). *Computer Vision*. Prentice Hall. ISBN 978-0-13-030796-5.
- [16]. Tim Morris (2004). *Computer Vision and Image Processing*. Palgrave Macmillan. ISBN 978-0-333-99451-1.
- [17]. Bernd Jähne; Horst Haußecker (2000). *Computer Vision and Applications, A Guide for Students and Practitioners*. Academic Press. ISBN 978-0-13-085198-7.
- [18]. Dana H. Ballard; Christopher M. Brown (1982). *Computer Vision*. Prentice Hall. ISBN 978-0-13-165316-0.
- [19]. Latombe, Jean-Claude (2012). *Robot Motion Planning*. Springer Science & Business Media. ISBN 978-1-4615-4022-9.
- [20]. *Planning Algorithms*, Steven M. LaValle, 2006, Cambridge University Press, ISBN 0-521-86205-1.
- [21]. *Principles of Robot Motion: Theory, Algorithms, and Implementation*, H. Choset, W. Burgard, S. Hutchinson, G. Kantor, L. E. Kavraki, K. Lynch, and S. Thrun, MIT Press, April 2005.
- [22]. Mark de Berg; Marc van Kreveld; Mark Overmars & Otfried Schwarzkopf (2000). *Computational Geometry* (2nd revised ed.). Springer-Verlag. ISBN 978-3-540-65620-3. Chapter 13: Robot Motion Planning: pp. 267–290.
- [23]. Donald L. Pieper, *The kinematics of manipulators under computer control*. PhD thesis, Stanford University, Department of Mechanical Engineering, October 24, 1968.
- [24]. Lynch, Kevin M.; Park, Frank C. (2017-05-25). *Modern Robotics*. Cambridge University Press. ISBN 978-1-107-15630-2.
- [25]. Siciliano, Bruno; Khatib, Oussama (2016-06-27). *Springer Handbook of Robotics*. Springer International Publishing. ISBN 978-3-319-32550-7.
- [26]. Paul, Richard (1981). *Robot manipulators: mathematics, programming, and control : the computer control of robot manipulators*. MIT Press, Cambridge, MA. ISBN 978-0-262-16082-7.

- [27]. J. M. McCarthy, 1990, Introduction to Theoretical Kinematics, MIT Press, Cambridge, MA.
- [28]. J. J. Uicker, G. R. Pennock, and J. E. Shigley, 2003, Theory of Machines and Mechanisms, Oxford University Press, New York.
- [29]. Paul, Richard (1981). Robot manipulators: mathematics, programming, and control : the computer control of robot manipulators. MIT Press, Cambridge, Massachusetts. ISBN 978-0-262-16082-7.
- [30]. John J. Craig, 2004, Introduction to Robotics: Mechanics and Control (3rd Edition), Prentice-Hall.
- [31]. J. M. McCarthy and G. S. Soh, Geometric Design of Linkages, 2nd Edition, Springer 2010.