Path Planning of a Robotic Manipulator for 3D Scanning of Moving Object on a Conveyer

Dugganna Ralalage Buddhika Kapila Kumara Weerasekara

(178671J)

Thesis submitted in partial fulfillment of the requirements for the degree Master of Science in Industrial Automation

Department of Electrical Engineering

University of Moratuwa

Sri Lanka

August 2021

DECLARATION

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date:

The above candidate has carried out research for the M.Sc. thesis under my supervision.

Signature of the Supervisor(s): Date:

Prof. A.G.B.P. Jayasekara

Dr. K.J.C. Kumara

Abstract

Robot manipulator systems are used in an industrial automation system to minimize human effort or involvement and increase product or service quality. Generally, automated robot systems are used to perform material handling, assembly operations, and quality inspection of the manufacturing system to achieve better performance in the precision, accuracy, and production rate of the continuous operation. In the injection molding industry, the surface quality of the output product is critical for producing a high-quality product. The surface scanning method is a common method for inspecting product surface quality. Most of the existing scanning methods need to identify and place the object in a fixed and known position to do quality scanning. In this research, we proposed a method of using a 4 DoF robot manipulator to move the scanner to get a quality surface scanning output for injection-molded products which are moving on a conveyor. Robot manipulator path planning is one of the main objectives of the research. The movement of the robot's end-effector must change depending on the object's orientation. The object image feature extraction method is used to determine the orientation of the object.The angle value's maximum accuracy ranges from $+5^0$ or -5^0 .The robot's end effector angle is changed according to the measured orientation of the object. The robot end effector is required to follow the object without any relative speed on the conveyor and maintain the absolute maximum speed to achieve an effective scanning output.According to experimental results, the optimal conveyor speed is $10cms^{-1}$.The speed control system is used to maintain the conveyor speed without any external disturbance and measure the speed, and it is fed to the robot system to maintain the relative speed of the conveyor. The distance between the scanner and the object is measured using an ultrasonic sensor. This sensor feeds the distance to the system, and the distance helps to maintain the path trajectory and takes into account the quality of the scanning output. A portable 3D scanner, the Quanser Kinova 4 DoF robot arm, and the MATLAB Simulink platform are being used to simulate the proposed system.According to the results, the conveyor speed was set at 10 cms^{-1} and the robot end effector moved on the trajectory based on the object orientation angle.

***Keywords*-Robot Manipulator, 3D Scanning,Path Planning,Image Feature Extraction.**

ACKNOWLEDGMENTS

It is with great pleasure that I acknowledge the assistance and contributions of all the people who helped me to successfully finish my Master's thesis.

First, I extend my sincere gratitude to my research supervisors, Prof. Buddhika Jayasekara and Dr. K.J.C Kumara, who provided me with their continuous support and assistance throughout the course of this thesis. Without their advice and encouragement, this thesis would never have been accomplished. I also would like to thank my progress review committee member, Prof. Chandima Pathirana, for delivering their guidance and valuable comments for successfully continuing my thesis work throughout the past year.

Also, my appreciation goes to the staff and all of my friends in the Department of Mechanical and Manufacturing Engineering and Department of Engineering Technology, University of Ruhuna, for their invaluable support, especially in taking part in the conducted experiments.

Finally, I would like to extend my deepest gratitude to my family. The blessings of my mother, father and brothers undoubtedly helped me in making this endeavor a success.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Introduction

Industrial robot systems have been developed mainly for automated material handling, assembly operations, and quality inspection processes. All tasks that human workers can do are processed by robots preciously, accurately, and at a speed greater than while operating 24 hours at a minimum. With the integration of the conveyor system, a complete set of operations has been realized within a manufacturing cell. Though a robot can move into 3D space and pick and place or scan the object and do assembly, for instance, the overall robot performance and output of the manufacturing cell can be improved when several operations can be achieved simultaneously.

A robot manipulator with a conveyor system to grasp an object with a known orientation reported in the literature developed a simple algorithm for minimization of relative velocity between the robot end-effector and the object (i.e. the same as the conveyor speed). The final output of the injection molding product is placed on a conveyor at various positions and orientations. Therefore, the scanning process is needed to identify the surface quality of the product considering the object positions.

Detecting objects and their orientation necessitates the use of sensors or camera systems. Also, such designs are limited to certain objects or shapes. This research work addresses how a robot can perform such a task while objects that are in

Figure 1.1: Different types of task achieved use in Robot Manipulator [1] Painting Operation,[2] Scanning Operation, [3] Pick and Place Operation.

various orientations (in 2D) are moving on a conveyor at a known or measured speed. Various tasks required for simultaneous operations can be integrated into the main motion plan design by extending the demonstration here with a 3D scanning focus.

1.2 Research Background

This research is based on the industrial real-time problem. According to the proposed system, we can replace a human with a robot manipulator in the quality inspection of the production line. At this time, the majority of applications and systems in the industry are available for scanning and inspecting the quality of the products on the production lines. Many systems are needed to fix the target object in the correct known place and start to inspect the properties of the object. Therefore, it is not suitable for the large-scale manufacturing industry. The reason is that processing time is very important for the production line. Then we tried to develop a system to implement that production line to minimize the quality of the inspection time. The 4DoF robot manipulator is the main part of the system. Furthermore, the conveyor is linked to the robot manipulator, and the speeds of the conveyor and robot end-effector are regarded as the system's effective scanning output.

1.3 Problem Statement

This study focuses on the product quality inspection of injection molded products.In the production industry, the main problem is how to check the product quality without any traditional method or human involvement and how to minimize the quality inspection time. The surface finishing and object shape are primarily scrutinized during the quality inspection process of injection molding products.Therefore, the manufacturer should inspect their product from time to time. It is a very difficult process as well as a time-wasting process. Therefore, manufacturers are moving to use modern technology as a robotic system in their plants to do that process. This research mainly focuses on robot manipulator path planning for moving an object on a conveyor system to minimize human involvement in the object scanning process. Furthermore, the speed changes of the conveyor and maintaining the effective distance between the object and the scanner have to be considered by such a robotic manipulator system.

1.4 Novelty of the Research

The target object should be placed in a known and fixed position in some of the existing object scanning methods. [1]. In this case, the target object is moving on a conveyor, and the scanning process is initiated during that moving period. Most of the "moving object pick and place operations" are used to regularly shape the target objects [4]. It may have a cylindrical or conical shape. Therefore, object orientation is no need to identify. For this study, we will primarily focus on non-regular objects, and must correctly identify the object's orientation. Therefore, we use the object feature extraction method to detect the object orientation.

1.5 Objectives

- Generate the object's 3D model and find the robot's moving path using the manual method.

  The robot manipulator needs to identify the way-point for each 3D object. A 3D model must be created in order to generate the way-points.And also use some CAD software to generate the slices and way-points for the target objects.

- Find the optimum conveyor speed for an error-free scanning trajectory, following the 3D model's generated way-points.

  This system mainly consists of a robot manipulator and an object-moving conveyor.And it needs to be synchronized with the end-effector speed and conveyor speed to achieve error-free scanning. The end-effector speed must be calculated using the 3D model trajectory.

- Simulation of path planning with a PID controlled conveyor system and forward kinematics and inverse kinematics of the 4DoF robot manipulator.

  The conveyor is controlled by a PID controller using the computed optimum speed. Robot manipulator system kinematics are modelled using the Robot system toolbox, and speed control DC motor simulation is employed to simulate the system.

- Object orientation detection generates the robot's end-effector moving path and changes the scanner's holding angle.

  A camera system is used to take an image of the object's current position and it is used to analyze the object's orientation compared with the reference image.The angle of the object with respect to the reference image angle is calculated using a feature extraction method.

1.6 Thesis Overview

The overview of this thesis can be summarized as following,

- Chapter 1 - Introduction.
- Chapter 2 -Literature Review for scanning method and robot manipulator trajectory generation.
- Chapter 3 -Methodology for the proposed system implementation.
- Chapter 4 -Experimental Setup for proposed system with including Robot manipulator with conveyer.
- Chapter 5 -Results and Discussion.
- Chapter 6 -Conclusion and Future works.
- Chapter 7 -Publications and Reference.

1.7 Research Limitation

The changes in environmental lighting directly affect the scanning output. Therefore, we assume the lighting condition of the scanning environment will remain unchanged during the scanning period. Automatic scanning will be conducted only for objects that are possible with respect to surface colour and complexity of fine details with a high quality manual 3D scan.

Chapter 2

LITERATURE REVIEW

2.1 Robot Motion Representation

The kinematics of the rigid body in a 3D space is the most impotent section of the robot manipulation. Let's consider F_w as a reference frame and O_w as the origin of the system. Then the rigid body frame can be defined as X_w, Y_w, Z_w. Also, translation and rotations shall be used to represent the relationship between two frames [2] shown in Figure 2.1

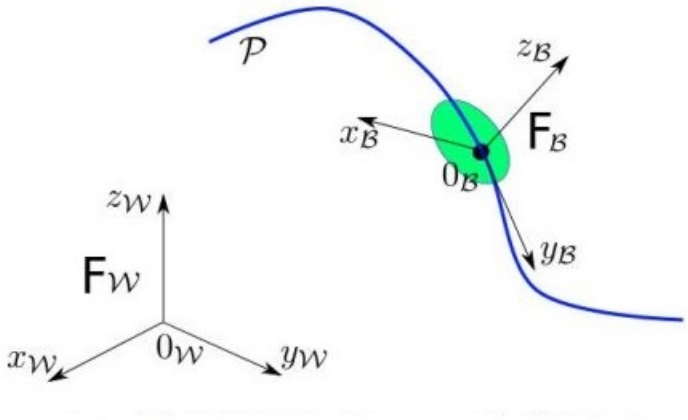

Figure 2.1: Rigid body represent in 3D space

In the case of a robot manipulator, a rigid body transformation matrix may include each link. mainly consider the translation and rotational matrix for the robot manipulator system. But the thing is, in various robot manipulator ap-

plications, end-users can not be involved in determining complete functions for the robot manipulator motion. Therefore, the motion path should be created in the robot program using limited position and orientation information as well as a set of way-points along the considerable path. Therefore, the trajectory path is used to move the robot end-effector through the way-point which generates the path trajectory. The trajectory refers to the time history of position, velocity, and acceleration for each degree of freedom in Cartesian space or joint space in a robotic manipulator [2].

The kinematics of the manipulator must be solved in order for the manipulator to move correctly. The kinematics of robot manipulators have gotten a lot of attention so far. The DH approach and the development of the Robotics toolbox on the MATLAB platform were introduced by Peter. I. Corke. This toolbox allows for forward and inverse kinematics analysis as well as simple kinematics simulation of the robotic arm. The DH method and the product of exponential formula were used to create a kinematics model of a 4-DOF robotic arm, and it was proven that these two approaches produce the same answer for the robot manipulator. [3].

2.2 Trajectory Planning and Path Trajectory Generation

Both position and orientation shall include the motion of the robot manipulator. Then consider the 4 DoF robot manipulator, which we need to control the position of the end effector in 3D space. Therefore, the robot manipulator requires 3 DoF to control the position of the end effector in space. The rotation of the last joint of the robot is the remaining degree of freedom of the end effector. Trajectories are time functions defined in geometrical space, essentially Cartesian space and joint space.

In most cases, the manipulator needs to specify the movement of the robot

using a detailed way-point of the task. Then the users need to provide a sequence of way-points. The sets of waypoints should include the representation of the starting point, intermediate point, and endpoint of the task. Each set of way-points can include different information depending on the design task. Each way-point of the Cartesian space describes the position and orientation of the manipulator. as well as joint space, describes the represent of all joint angles. The following motion conditions as a triplet associating the position, velocity, and acceleration at time t along the trajectory [4].

For Cardician space,

$$P_{(t)} = (X_{(t)}, V_{(t)}, A_{(t)}) \tag{2.1}$$

For Joint Space,

$$P_{(t)} = (Q_{(t)}, \dot{Q}_{(t)}, \ddot{Q}_{(t)}) \tag{2.2}$$

There are different kinds of motion planning techniques that are used to create smooth motion and control the trajectory of the robot system. The continuous trajectory function and continuous first-order derivative will generate a smooth trajectory. A continuous second-order derivative is desirable for a minimum jerk trajectory. Minimum jerk means it minimizes the wear on the robot arm joint mechanism.

When considering the smooth motion between two consecutive waypoints, it may satisfy the following constraints according to position, velocity, and acceleration. X_0 and X_1 are two consecutive way pints of smooth motion trajectory.

$$X_{(t=0)} = X_0 \qquad \dot{X}_{(t=0)} = V_0 \qquad \ddot{X}_{(t=0)} = A_0 \tag{2.3}$$

$$X_{(t=1)} = X_1 \qquad \dot{X}_{(t=1)} = V_1 \qquad \ddot{X}_{(t=1)} = A_1 \tag{2.4}$$

V_0, A_0, V_1 and A_1 constraints are necessary to smooth trajectory operation between the waypoints, and A_0 and A_1 are an effect of the jerk. Then, for a minimum jerk and smooth trajectory, these values should be zero for the initial and final waypoints. The Quintic Polynomial formula is required to find the above constraints [4].

In order to plan minimum-jerk motions for surface-mount assembly robots, a fifth-order polynomial motion model was investigated. Smooth transitions between global route segments can be achieved with such a motion model, resulting in high precision part placement accuracy. The tradeoff for this accuracy is cycle time. The coordinated planning and control technique has been improved to allow more realistic motion models to be used. The link between the average and maximum speeds enabled the method to do nonlinear speed planning in the fifth-order minimum-jerk polynomial motion model. The algorithm can be expanded to incorporate more general polynomial motion models, as well as any motion model that can be defined analytically and whose average/maximum speed relationship can be calculated. The generalized method and the minimum-jerk motion model were shown to be useful in a dual-robot SMT assembly machine experiment [5].

2.3 Pick-and-Place of Dynamic Objects

Mobile robot manipulators have brought a modern level of flexibility to traditional automation tasks such as pick and place robot manipulation, but are not yet capable of the same speed and reliability as industrial automation. In pick and place of dynamic objects, the researchers present approaches to 3D

perception and manipulator motion planning that enable the general purpose of the robotic manipulator platform to recognize and manipulate a variety of objects at a rate of one pick-and-place operation every 6.7 s, and work with a conveyor belt carrying objects at a speed of 33 cm/s [6].

Kinematic planning for robotic arms has been demonstrated to be capable of planning for robust manipulation of static objects. However, this approach falls short when manipulating moving objects, such as picking up a conveyor belt. The test of cautiously getting moving items is that these activities require movements that don't include enormous deceleration, to avoid jerking the object, as well as figuring out the proper time at which the object can be picked up. prepared to do so cautiously, getting the item at the most punctual possible point in its trajectory. To battle the high-dimensionality of the time-defined kinodynamic planning issue, our approach employs informative heuristics and adaptive dynamic motion primitives.

In one of the research exhibited pick-and-spot tasks performed by a PR2 at a pace of 6.7 s per object at a 91% achievement rate. A Comparative procedure on a moving work surface yielded an 87% achievement rate [6]. opportunity to get better remaining parts in the areas of system integration and end-effector customization. The speed at which the PR2's arms can move is demonstrated to be a restricting element in framework execution, and the gripper configuration was not especially suitable for retaining the effect of moving articles. Regardless of these mechanical constraints, the PR2 proved to be capable of responsive, high-throughput object manipulation.

2.4 Kinodynamic Motion Planning

The kinodynamic motion planning problem is to pick up a moving object such as a bottle, cup, or glass on a conveyor belt. Based on the above knowledge

of the object trajectory, we need to manually select the grasp of the object. We divide the motion plan into three phases, which are differentiated based on the pose of the end-effector with respect to the object.

2.4.1 Reaching

The end-effector isn't near the object, and the controller is moving to position the end-effector at the object.

2.4.2 Grasping

The end-effector is moving from near the object to contact with the object, and tracking it to secure its hold on to the object. In this phase, the manipulator needs to keep the relative position of the end effector and the object the same, otherwise, the object will slip out of grasp.

2.4.3 Lifting

The end-effector has achieved a stable grasp of the object, which allows it to move the object as if it were rigidly attached. Here, the manipulator would be free to move in space, or it could be constrained by the special dynamics of the object, such as a fluid-filled container not being jerked.

Figure 2.2 shows the reaching, grasping, and lifting of the robot end effector in a given specific task [4].

2.5 Moving Object Tracking and Orientation Detection

A video sequence consists of several frame sequences which have a certain surplus of continuity. The detection of a moving object in a video is conducted in

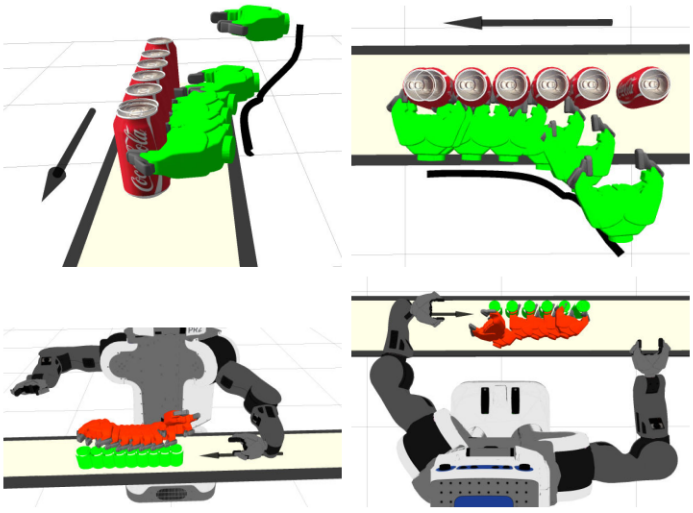

Figure 2.2: Example gripper trajectories as generated by the adaptive motion(Reaching, grasp, and lifting)

a way that frame sequences are extracted from the video sequence according to a definite cycle. Therefore, moving object detection in still images only moving object tracking is more reliant on the moving characteristics of objects, i.e., the continuity of time, which is the difference between moving objects in a frame and object detection in frames.

To acquire an image using MATLAB, a video input object which represents the connection between MATLAB and the image acquisition device. Normally, it requires a certain video processing algorithm to prepare the image for further analysis. In some research, the video is analyzed as it is viewed. To analyze color frames, it is necessary for the first segment.

One of the most commonly used image segmentation methods in object detection is edge detection. Since edges contain some of the most useful information in an image.It can be used to extract the boundaries of each different object in an image. In grayscale images, an edge may be defined as the local discontinuity in the intensity values that exceed a given threshold. For color frames, similar edge detection methods can be applied to the intensity component of the image.

2.5.1 Feature detection

In computer vision and image processing, the concept of feature detection refers to methods that aim at computing abstractions of image information and making local decisions at every image point whether there is a frame feature of a given type at that point or not. [7]

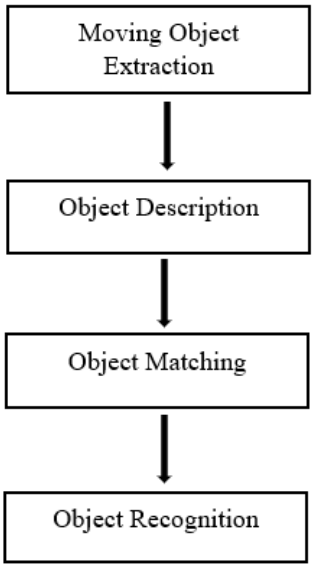

Figure 2.3: Object Recognition Block Diagram

Figure 2.3 shows a basic object recognition diagram to recognize each object in an image. Each object in an image will have a different shape and size. All images of shape and size are stored in the MATLAB algorithm as library templates. By comparing the results of the shape and size of the image produced by MATLAB with the stored value in the library templates, objects in a frame can be recognized. [7]

Figure 2.4 shows a flow chart that explains the basic algorithm for tracking simple objects (segments of the segmentation algorithm) in a video image. The detection of an object and its movement is an initial process for tracking. The

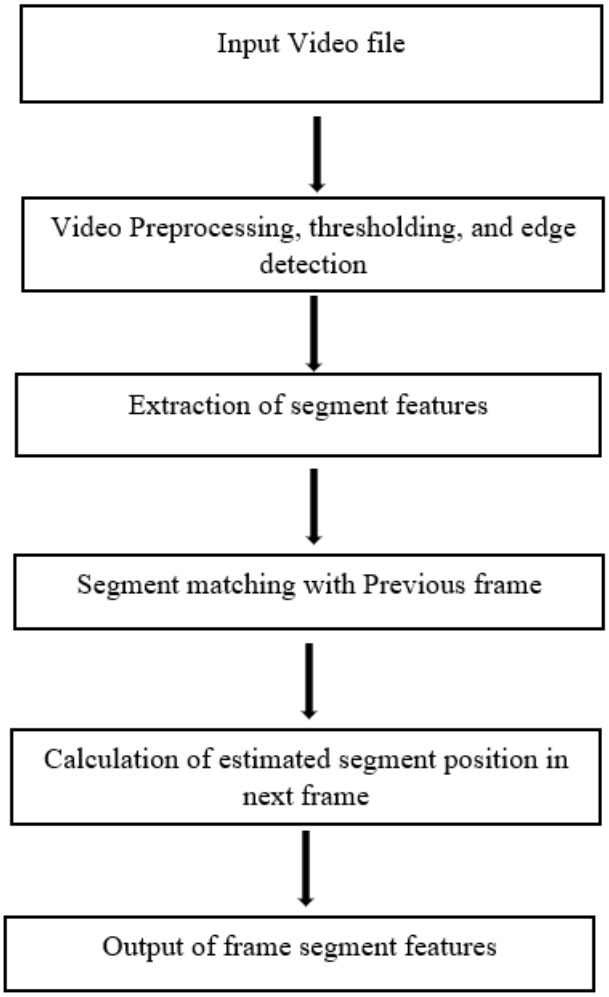

Figure 2.4: Basic Algorithm for Tracking of Simple Object in a Video Image

tracking must be supported by additional methods for clear-cut object classification. It is required for several reasons. A detected object in one frame and a separately detected object in another frame can mean another object.

Another study applied an object tracking algorithm to the modified film, resulting in trajectories for 14 moving vehicles in two different scenarios. The feature point recovery was not implemented in the first situation, but it was implemented in the second case according to the proposed method of regeneration. The relevant reference trajectories' initial trajectory point co-ordinates were assigned the same values. Then we compared the results to the reference trajectories and derived two performance metrics: a percentage of correctly tracked objects and a tracking deviation. If the distance between a moving object's position in the generated trajectory and its position in the reference trajectory does not exceed the provided maximum distance threshold for each video frame, it is regarded as successfully tracked. The maximum distance criterion was manually set to a percentage of the object's bounding box size. The proportion of successfully tracked objects was calculated by dividing the total number of reference trajectories by the number of successfully tracked objects. The tracking deviation was calculated as the average standard deviation of all successfully tracked objects' resultant trajectories from the reference trajectories. [8].

2.6 3D Scanning

3D scanning is one of the most widely used processes for capturing the shape of an object using a 3D scanner. The data collected by the 3D scanner can be used to build a digital 3D representation. The scanning results of the 3D file of the object can be saved, edited, or built-up the 3D object using 3D printers.

There are so many types of 3D scanners and 3D scanning techniques or technologies used in the presence. The 3D scanner can be divided into two main types. The short-range 3D scanner is used as a laser-based technique to scan objects. The laser-based 3D scanner uses a process called trigonometric triangulation to accurately capture a 3D shape as millions of points.

Figure 2.5: Handheld 3D Laser Scanner

The long-range 3D scanner can be divided into two major forms. That's called pulse base and phase shift. Both two forms are used for large objects such as structures, buildings, and also military vehicles.

The Manual Scanning method shown in Figure 2.5 uses a single laser point or laser beam to scan around the target object to be scanned. The working principle of the laser beam scanner, first the beam is made to fall on the object surface and as it reflects off of the 3D scanned object, the change in its trajectory is recorded by the sensor [1].

Therefore, considering the technique of 3D scanning, we can classify the 3D scanning techniques shown in Figure 2.6.

2.6.1 3D Scanning for Quality Inspection

In the production industry, laser scanners are used to check their production quality. Most scanners are usually manually operated. Therefore, the process of quality inspection is very expensive. Some manufacturers use an automated measuring system for a part having a freeform surface. Appropriate hardware and software systems are required to implement the automated measuring process. Most of the time, laser scanners are used as the main hardware part of

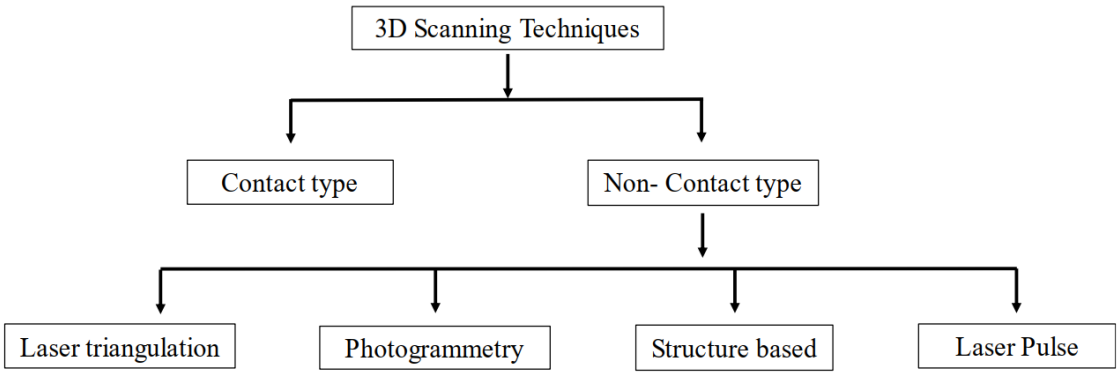

Figure 2.6: 3D Scanning Techniques Classification

that system. But the thing is, if it is using a laser scanner, the scanning device and setup fixture provide the proper position and orientation for the object to be measured. There are many scanning parameters that should be considered in the generation of the optimum scan path, like view angle and depth of field. The length of the stripe. In that method, the measured data sets are stored automatically and the quality of each point is evaluated by considering the difference between their CAD models. [9].

Most of the time, Coordinate Measuring Machine (CMM) and three-dimensional (3D) laser scanners are used in the manufacturing field for quality inspection [9]. The trigger type CMM acquires point data by touching the probe to the object. And also, scanning type CMM can capture more sampling points without touching the surface.

Literature Summery with research gap

The DoF value is important for effective robot motion operation. Previously, the 7 DoF PR2 robots [6] were used in research for moving objects related to robot manipulator operations. For the scanner moving operation in this study, we used a 4 DoF robot manipulator.

The "Teach" method was used to load the paint data onto the robot ma-

nipulator in the preceding work reference, [4]. It must first move the robot's end effector along the desired path before recording all of the point data. In our case, we imported the previously calculated way point data produced by the 3D model.

There are many different methods of object orientation detection that are employed in various types of research. Some researchers use object-moving videos to extract an image frame from the video. They used an image capture tool in MATLAB to take the image from a video recording. After that, they used a MATLAB based image edge detection method to identify the object's position and orientation. However, in our case, we propose a method that takes the image directly from the camera and extracts the surface feature to detect the orientation in the shortest amount of time.

Chapter 3

METHODOLOGY

3.1 Trajectory Generation for Proposed System

This kind of system is mainly useful for the injection molding industry. The reason for this is that the surface finishing is more important than the product's quality.Therefore, the scanning process is based on the outer surface. Consider regular objects such as cylinders, cones, and spherical objects, each of which has a distinct surface shape.Therefore, it's easy to generate a trajectory based on their shape. However, for some irregular shapes, we must first identify the surface shape of the object and then use some method to generate a trajectory. In this research, the main objective is to generate an irregular object trajectory path using a 3D model which is generated from the manual scanning method.

3.2 Moving Path Generation for Regular Shape Object

In this case, we use a cylindrical, conical, and spherical shape as the regular shape of the object. Then we need to identify the object parameters like height and base radius for generating the surface path using a mathematical equation.

Figure 3.1 depicts the mathematical functions used to generate cylindrical, conical, and spherical-shaped object surface paths, which can be used as a trajectory path for regular object scanning. This type of function, however, can

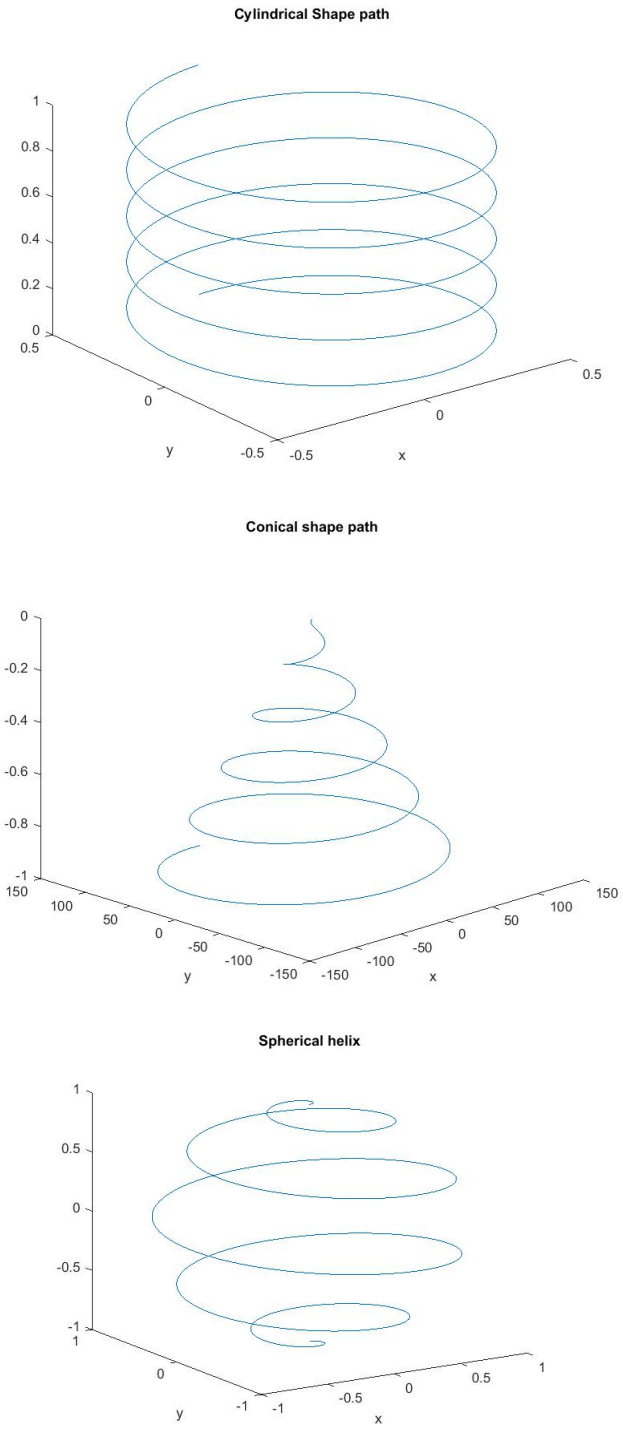

Figure 3.1: Reguler Path Shape for Cylindrical Conical and Spherical Object

not be used on non-regular objects. The reason for this is that the surface is not uniform. We need to use some method to identify the surface path trajectory to generate the waypoint to move the end effector of the robot manipulator.

3.3 Moving Path Generation for Non-Regular Shape Object

The term "non-regular object" refers to an item whose surface can not be described mathematically. Therefore, we need to use some method to identify the object's surface. Use the manual scanning method to generate a 3D model and it is used to identify the surface path way points for the system.

3.3.1 Manual Scanning Method

The 3D sensing scanner has some scanning limitations. When starting the scanning process, like in Figure 3.2, we should maintain the distance between the target object and the scanner. and also keeps the scanner's movement speed consistent with the manufacturer's instructions [10]. Keep the distance between the object and the scanner within the limits listed below.

The maximum and minimum distances between the object and the scanner are 38.1 mm and 152.4 mm, respectively.

3.3.2 Generate 3D Model

According to scanner limitations, such as speed and distance between the target object and the scanner, the scanner starts to move around the object's surface, as shown in Figure 3.2. After that, using the 3D Sense software platform, we can generate a 3D model like shown in Figure 3.3, and it can be saved as a (.STL) file type. An STL file can be easily imported into CAD software to generate the slice for taking waypoints.

Figure 3.2: Manual Scanning using Portable 3D Scanner

Figure 3.3: Genarated 3D model using Portable 3D Scanner

3.3.3 Generate the Path

After importing a 3D object to CAD software, define the planes of the object which are placed on the actual object. According to the plan, the object needs to be divided into many small slices. as well as defining the new plane for each slice and generating the path for each plane. Using CAD software, we can identify the coordinates of the points of each plane and define the waypoints for each plane. After that, using each waypoint, we can generate the point cloud of each trajectory. Then it can directly feed into the mathematical model and define the trajectory generation of irregular objects.

3.3.4 Irregular Shape Trajectory Generation

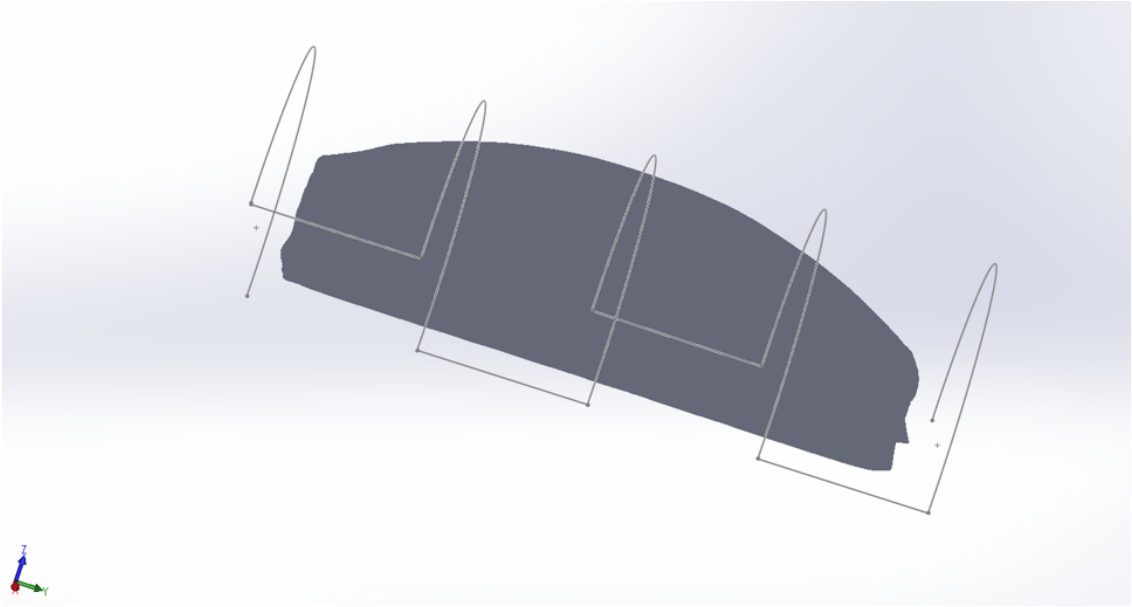

Figure 3.4: Import the 3D model into SolidWorks and Slice the equal Part

The Iraguler shape object trajectory generation process can be described based on the flow chart shown in Figure 3.5 Starting with the 3D model generation, a manual scanning method and the creation of a 3D model are required to account for path trajectory generation.After analysing the silces,the specific point needs to be identified for the tragectory genaration. We can generate the exact trajectory path for the given object after applying the trajectory's boundary conditions.

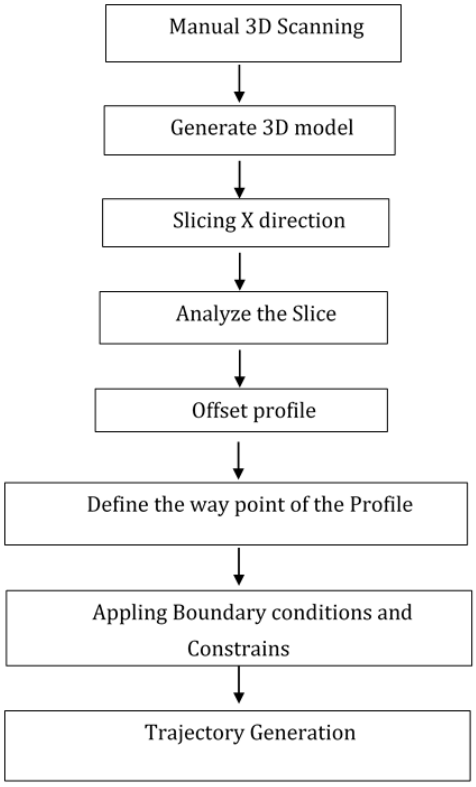

Figure 3.5: System Flow Chart for irregular shape Trajectory Generation

3.4 Kinematics Model for Proposed Robot system

3.4.1 Forward Kinematics

A Four-DoF robot manipulator is a set of serial connections that are connected from the base frame via the end-effector to each other. There are 4 joints with a serial link and the two-finger grips are used as an end-effector. Take a look at the forward kinematics based on the joint variables (length and angle).By using forward kinematics, we calculate the end-effector position and orientation. The Denavit-Hartenberg (DH) method is the most common method for describing robot manipulator kinematics.

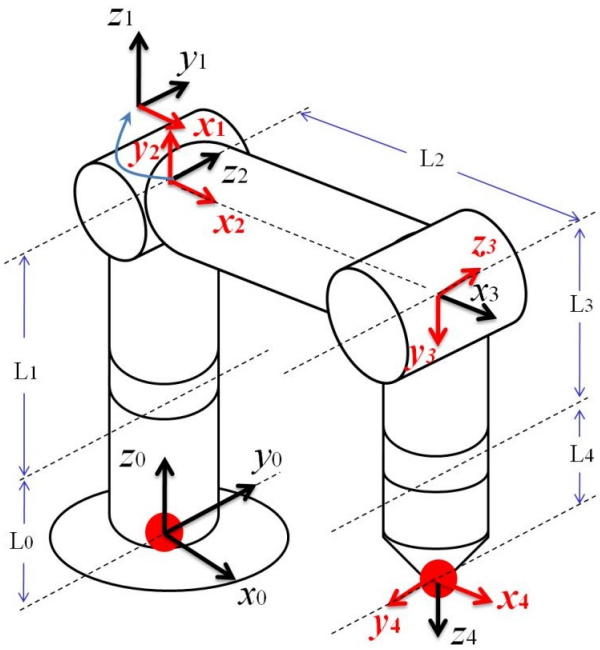

Figure 3.6: 4DOF Robot Manipulator Parameters

3.4.2 DH Method

The DH method is used to define the end-effector's position and orientation with respect to the DH parameters. These parameters are defined as a_{i-1}, α_{i-1} , d_i ,θ_i . The meaning of these parameters is the length of the link, the twist of the link, the offset link, and the angle of the joint respectively. In order to determine the DH parameters, a coordinate frame is attached to each joint. The Z-axis of the coordinate frame points along with the rotary or sliding direction of the joints.

α_{i-1} = the angle from Z_{i-1} to Z measured along X_{i-1}
a_{i-1} = the distance from Z_{i-1} to Z measured along X_{i-1}
d_i = the distance from X_{i-1} to X_i measured along Z_i
θ_i = the angle from X_{i-1} to X_i measured along Z_i

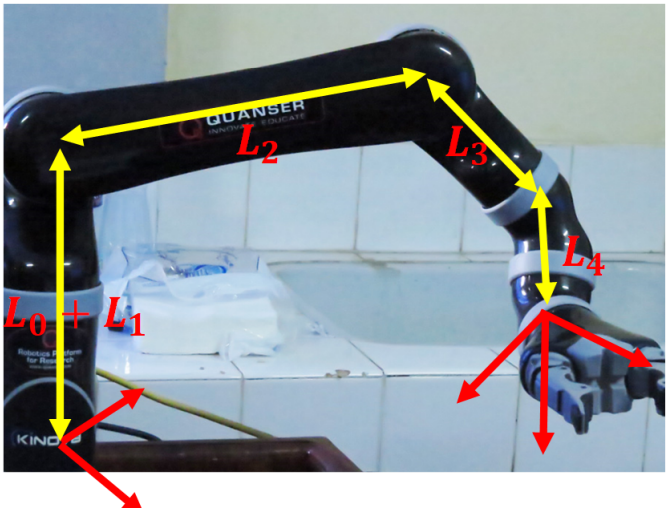

Figure 3.7: Quanser KINOVA 4DoF Robot Manipulator Link Length

3.4.3 Forward Kinematics Model for Proposed System

The 4 DoF robot manipulator is used in this study as a robot arm that is connected to the scanner and completes the motion trajectory, which provides information.The forward kinematics model is critical for determining the position and orientation of the end effector point.Using robot manipulator specifications (link length) and different angles of the link, we can generate the DH table and find the parameters of end-effector position and orientation. cite8316502.

Derive the individual link transform metrics and you can determine the portion of the matrices that represents the rotational and translation metrics.

$$L_0 + L_1 = 0.2755, L_2 = 0.29, L_3 = 0.1233, L_4 = 0.16 \tag{3.1}$$

Table 3.1: DH Table for 4DOF Robot Manipulator.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0.2755	θ_1
2	-π /2	0	0	θ_2
3	0	0.29	0	θ_3
4	-π /2	0	0.2833	θ_4

$$T_i^{i-1} = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & \alpha_{i-1} \\ S\theta_i C\alpha_{i-1} & C\theta_i C\alpha_{i-1} & -S\alpha_{i-1} & -S\alpha_{i-1} d_i \\ S\theta_i S\alpha_{i-1} & C\theta_i S\alpha_{i-1} & C\alpha_{i-1} & C\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

manipulator links angle set as following positions and end effector home position can be calculated using the above equations.

$$\theta_1 = (\frac{\pi}{2}) \ , \ \theta_2 = (\frac{-\pi}{6}) \ , \ \theta_3 = 0 \ , \ \theta_4 = 0$$

$$T_1^0 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.275 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

$$T_2^1 = \begin{bmatrix} 0.866 & 0.5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0.5 & -0.866 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

$$T_3^2 = \begin{bmatrix} 1 & 0 & 0 & 0.290 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

$$T_4^3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.283 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

$$T_4^0 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0.866 & 0 & 0.5 & 0.392 \\ 0.5 & 0 & -0.866 & 0.175 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

Then end effector position is, $x = 0m, Y = 0.393m, Z = 0.175m$

3.4.4 Inverse Kinematic

Inverse kinematics is the secret to functional robotic manipulator programming. The associated joint variables can be determined once the desired wrist frame is defined in terms of the base frame and these values are transferred to individual joint controllers to move the robot to the desired location.

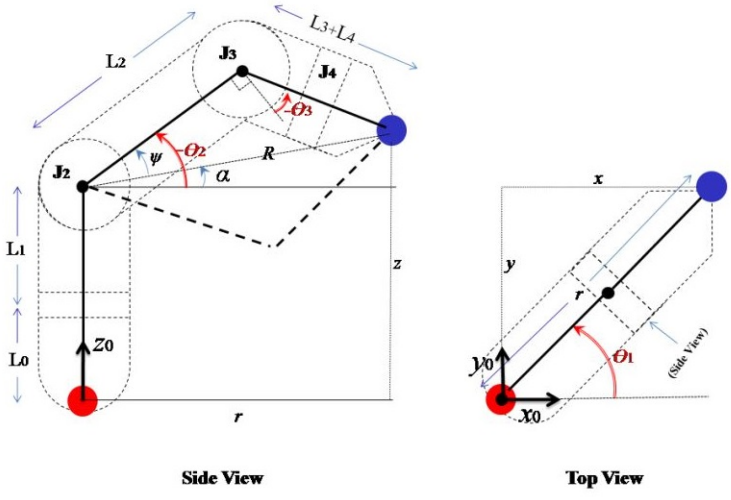

Figure 3.8: 2D top view, and side view schematics of the 4-DOF MICO robot arm

3.4.5 Inverse Kinematics Calculations for 4 DoF Robot Manipulator

We can calculate an inverse kinematics model for a proposed robot manipulator system using the values of Equation 3.1 and Figure 3.8.There are two sets of data taken from the inverse kinematics model. Based on the link parameters and other angles given in Figure 3.8, we can calculate two sets of solutions for the inverse kinematic model [11].

Solution set 1:

$$\theta_1 = \tan^{-1}(y/x) = 0 \tag{3.8}$$

$$\alpha = atan^2(z - (L_0 + L_1), \sqrt{x^2 + y^2})$$

$$\alpha = atan^2(0.7 - 0.2755, 0)$$

$$= 1.5708rad = 90^0$$

$$R = \sqrt{x^2 + y^2 + (z - (L_0 + L_1))^2} = 0.7 - 0.2755 = 0.4245$$

$$\varphi = \cos^{-1}(\frac{R^2 + L_2^2 - (L_3 + L_4)^2}{2.L_2.R})$$

$$= \cos^{-1}(\frac{0.4245^2 + 0.29^2 - (0.1233 + 0.16)^2}{2 \times 0.29 \times 0.4245})$$

$$\varphi = 41.63^0$$

$$\theta_2 = -(\alpha + \varphi) = -131.63^0 \tag{3.9}$$

$$\theta_3 = \sin^{-1}[\frac{L_2^2 + (L_3 + L_4)^2 - (x^2 + y^2 + (z - (L_0 + L_1))^2)}{2 \times L_2 \times (L_3 + L_4)}]$$

$$= \sin^{-1}[\frac{0.29^2 + (0.1233 + 0.16)^2 - (0.7 - 0.2755)^2}{2 \times 0.29 \times (0.1233 + 0.16)}] = -5.53^0 \tag{3.10}$$

Solution set 2:

$$\theta_1 = \tan^{-1}(y/x) = 0 \tag{3.11}$$

$$\theta_2 = -(\alpha - \varphi) = -48.37^0 \tag{3.12}$$

$$\theta_3 = -\pi - \sin^{-1}[\frac{L_2^2 + (L_3 + L_4)^2 - (x^2 + y^2 + (z - (L_0 + L_1))^2)}{2 \times L_2 \times (L_3 + L_4)}]$$

$$= -\pi - \sin^{-1}[\frac{0.29^2 + (0.1233 + 0.16)^2 - (0.7 - 0.2755)^2}{2 \times 0.29 \times (0.1233 + 0.16)}] = -\pi - 0.09 = -174.47^0 \tag{3.13}$$

According to the above calculations, we can define the robot manipulator's joint parameters for a given coordinate. In this research, we used the MATLAB software platform to design and implement the Inverse Kinematics model for a given task. All kinematics simulation results have been taken from the MATLAB Simulink model. To simulate the link configuration and joint configurations, we use the MATLAB robotics toolbox as well as results taken from using the same platform.

3.5 Minimum Error Scanning

For an effective scanning process, the scanner should be placed in the correct position with an accurate orientation. According to the proposed system, the robot end-effector moves through the desired trajectory based on the object's shape. The trajectory conditions change the distance between the object and the scanner from time to time during that process.Therefore, we need to measure that distance and try to maintain it based on the scanner's operating limitations.

3.5.1 Proposed Setup for Distance Measurement

The quality of the scanning process depends on the object's position and the distance between the object and the scanner. Then we need to maintain the distance between the object and the scanner accurately. Therefore, measure

the distance between the object surface and the scanner by using an ultrasonic distance measuring sensor shown in Figure 3.9. It can measure the distance and the Arduino microcontroller is used to process that distance value. If the distance comes to the desired limit, we can identify the quality of the scanning output. The MATLAB Arduino support package is used to process the distance value and it helps to connect the distance value to the Simulink model. The distance value is used as feedback for the robot manipulator, and its motion is affected by the distance.

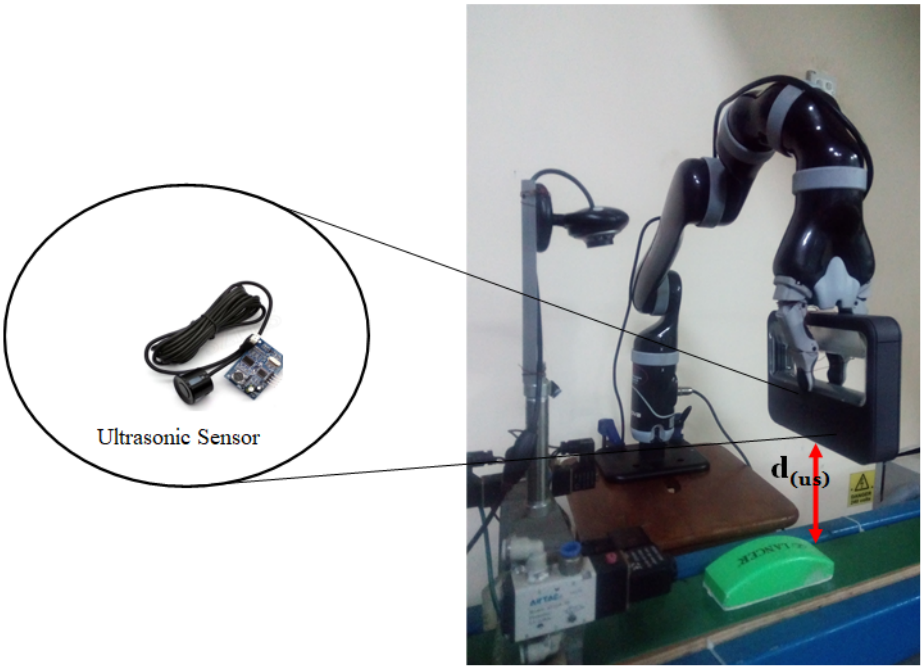

Figure 3.9: Distance Measuring Unit implementation

3.6 Object Orientation Detection

The position and orientation of the object are critical for obtaining a high-quality scanning output. We need to identify the correct position and orientation of the object placed on a conveyor. According to our experimental setup, the conveyor can be adjusted based on the size of the object. Therefore, we have no need to worry about the position of the object. To detect the object and its

location, we can use a capacitive proximity sensor.

The target object placed on the moving conveyor can have different orientations. Then, in order to obtain a high-quality scanning output, we must consider the object's orientation.Therefore, we use an image processing technique to detect that orientation. The high-quality camera module is used to capture the image of the target object. Figure 3.11 shows the different orientations of the target object, which is placed on a conveyor system. Following the capture of that image, it must be processed in accordance with the referral image.The image processing flow chat is depicted in Figure 3.10.

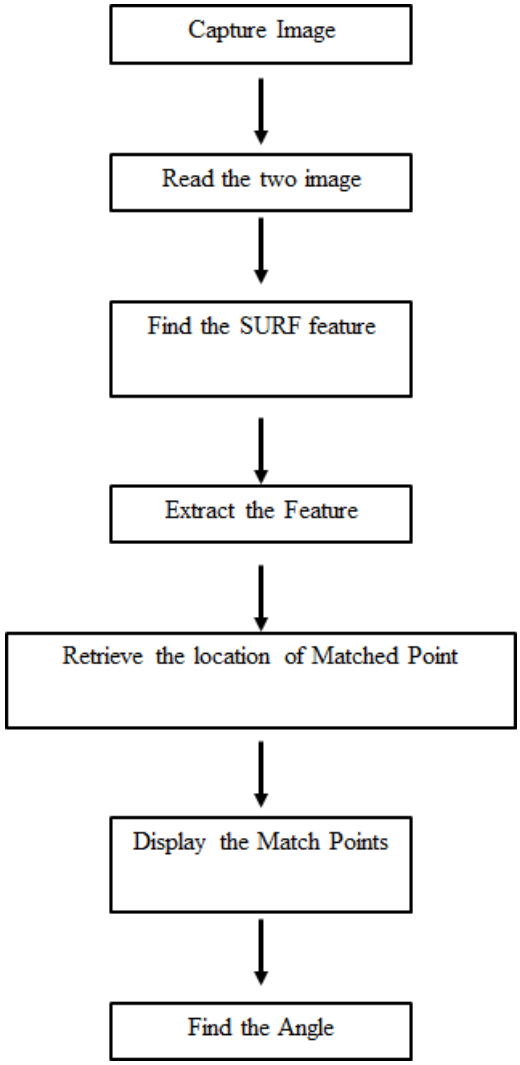

Figure 3.10: Orientation Detection Method flow chart

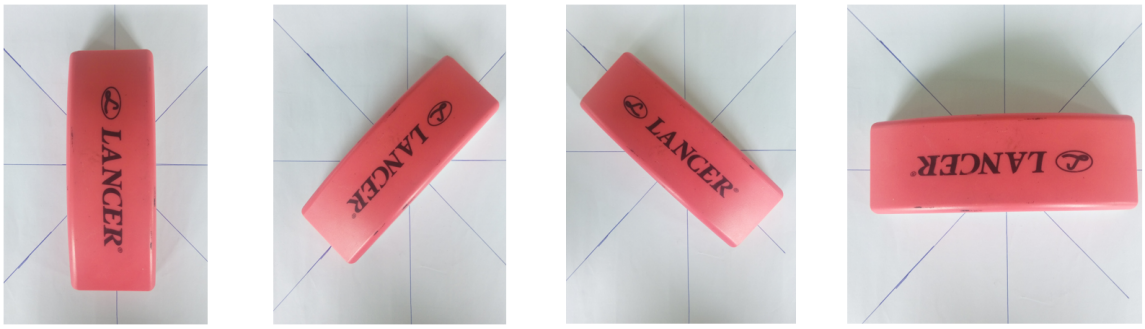

Figure 3.11: Different known angle for Object orientation detection

According to Figure 3.11, the first image (0^0 or 360^0) is considered as a reference image or template. The X-direction of the object is considered as the conveyor's moving direction. Then the reference image should be set as the conveyor's moving direction. The images taken from the camera module are imported into the MATLAB program [Appendix E], and it starts to process the feature extraction. [12]

Figure 3.12 depicts the feature extraction process. According to the feature extraction method, the angle will automatically be generated based on the feature variation.

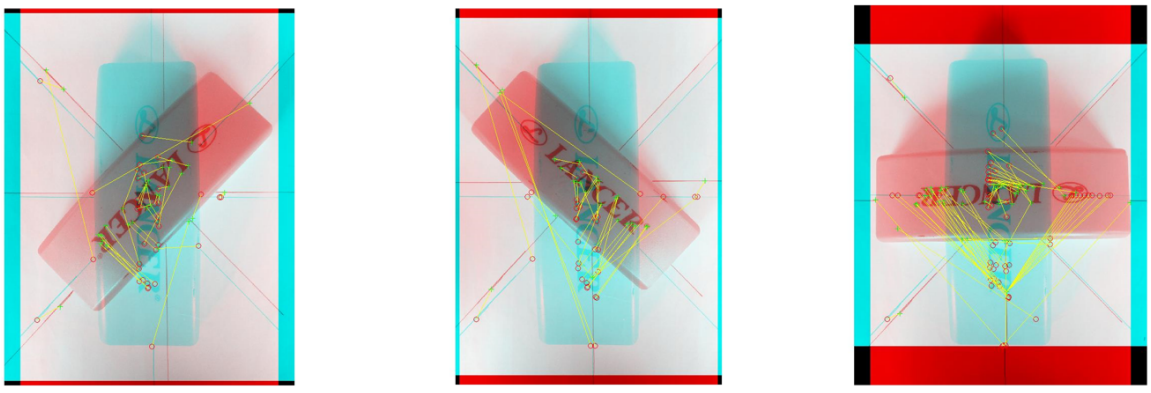

Figure 3.12: Find the surface Feature and extracting the features

Chapter 4

EXPERIMENTAL SETUP

4.1 Basic System Architecture of the Robot System

The Robot Manipulator Control system is the most important part of this research. Therefore, we mainly consider the basic robot manipulator system architecture. Many field devices connect with this system as shown in Figure 4.1 and need to get more details about the behavior of that system. According to the proposed system, there are three field devices that connect with the main controller. The input data and signals are processed using the QuaRC compiler and the MATLAB Simulink toolbox. And finally, feed the data to the Quencer 4 DoF Robot Manipulator to start the operation.

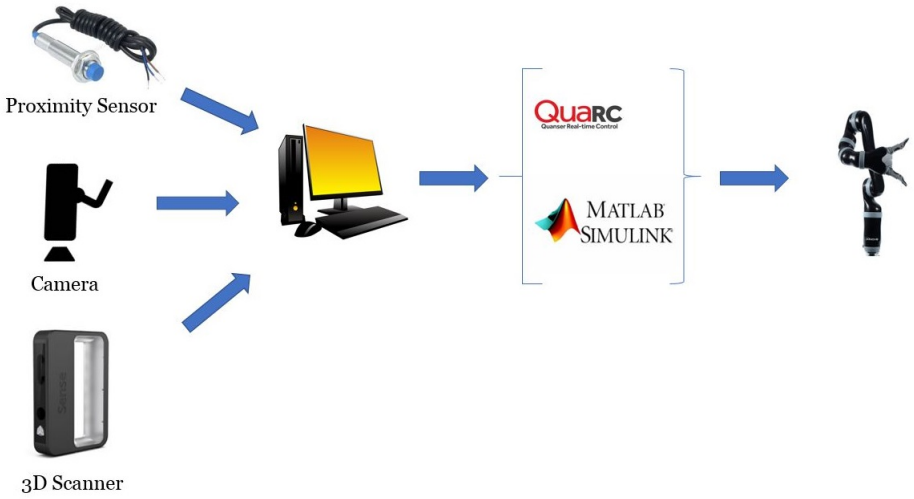

Figure 4.1: Basic Robot Manipulator Control System Architecture

4.1.1 Camera Module

The camera module shown in Figure 4.2 is used to take an image of an object placed on a conveyor. This module is directly connected to the system computer and the MATLAB Simulink platform and it captures the object image based on the Simulink block command. Also, the captured image was used to extract the features. Mainly, we need to consider the resolution of the camera to take a clear image. Figure 4.3 depicts the various images captured by the camera module.

Figure 4.2: Webcamera module

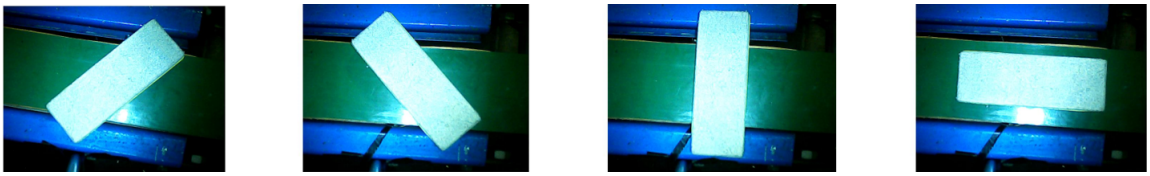

Figure 4.3: Deffetent orientations of the object captured form webcam.

4.1.2 3D Scanner

Figure 4.4: 3D sense Portable 3D scanner

The 3D scanner is the main part of this system for scanning target objects. After the scanning process, it will automatically generate a 3D model of the object. The 3D sense portable 3D scanner is used for this process and is shown in Figure 4.4. Before starting the automatic scanning process, we need to scan that target object manually and generate a 3D model as shown in Figure 4.5 and it can be imported into the CAD software. As a result, before we begin manual scanning, we must first determine the scanner's limitations and its range.

Figure 4.5: Three-dimensional image of the object

4.1.3 Ultrasonic Sensor Model

It provides a stable and precise distance measurement from 2 cm to 450 cm in these types of ultrasonic distance sensors and also has a concentration of less than 15 degrees and a precision of about 2 mm. The actual sensor sends out an ultrasonic sound which has a frequency of about 40 Hz. The transmitter and receiver are the main parts of the sensor. A transducer generates an ultrasonic sound, and the receiver detects the transducer's echo.The sound travels at a rate of approximately 340 ms^{-1}.This corresponds to about 29.421 milliseconds per centimeter. We can use this formula to measure the distance the sound has traveled.

$$Distance = (Time \times SoundSpeed) \tag{4.1}$$

But Sound has to travel back and forth. Therefore, the time should divide into two equal parts.

$$Distance = (\frac{Time}{2} \times SoundSpeed) \tag{4.2}$$

By using the above equation, the distance between the scanner and the target point can measure as follows, Echo Time = T_0

$$Distance(d_{us}) = \frac{T_0}{2} \times \frac{1}{29.42} \tag{4.3}$$

4.1.4 Ultrasonic Sensor Implementation

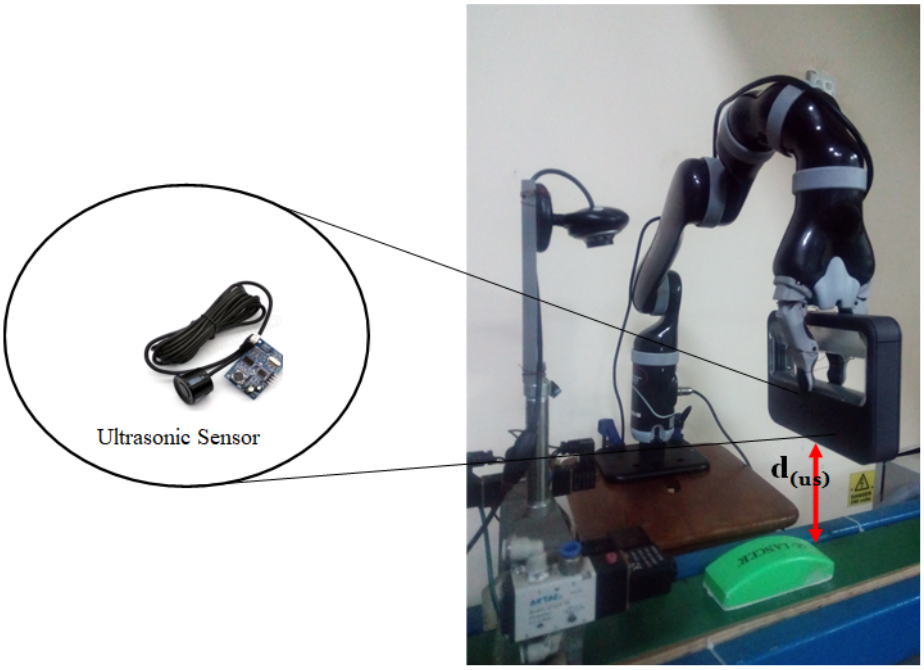

Figure 4.6: Distance Measuring Unit implementation

A distance measuring sensor should be placed on the robot gripper perpendicular to the target surface. The reason for this is that in that orientation, an accurate distance reading can be recorded.Therefore, we used a low-cost waterproof Ultrasonic distance measuring sensor to connect that experimental setup. The sensor is directly connected to the Arduino microcontroller, and it helps to calculate the distance between the object's surface and the robot manipulator

gripper. The target distance (d_{us}) is saved in the MATLAB script, and that MATLAB workspace is linked to the Simulink model. To achieve an accurate scanning process, the robot's trajectory will change based on the distance.

4.2 Conveyor System

Conveyor behavior is critical for obtaining high-quality scanning output as well as for trajectory tracking. The conveyor's moving direction, moving speed, and surface friction between the motor and the conveyor belt directly affect the scanning process. We assume the friction between the motor and the conveyer is negligible, but the object can slip off the conveyer belt.

If you get a quality scanning output without any jerking of the robot manipulator, the conveyer should move at a desired constant speed, according to the scanner requirements. Then, the relationship between the conveyor speed and the robot's end-effector speed is directly affected by the quality of the scanning process. The reason for this is that the relative velocity between the scanner and the conveyor should be kept as low as possible (=0) for that operation.Then, they mainly control the conveyor speed according to the scanner requirements and, from time to time, measure that conveyor speed and set the constant speed by using a PID controller. One of the main objectives of this research is to find the optimum conveyor speed. Before changing the speed, the conveyor's moving direction should be aligned with the X direction of the robot manipulator.

4.2.1 Optimum Conveyor Speed

The optimum conveyor speed is the same as the optimum end-effector speed. The Quintic formula is used to calculate manipulator end-effector position, velocity, and acceleration. The reason is that the robot manipulator has three positions, three velocities, and three accelerations. According to the Quan-

tic polynomial function, robot movements can be represented by the following equations.

Quintic polynomials cardician Space,

$$P(x) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{4.4}$$

$$P(y) = b_0 + b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4 + b_5 t^5 \tag{4.5}$$

$$P(z) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5 \tag{4.6}$$

Consider the two-point parameters of end effector movement, first point and last point define as ‘ a ’ and ‘ b ’.

$$SpeedRange = \begin{Bmatrix} V_{ax} & to & V_{bx} \\ V_{ay} & to & V_{by} \\ V_{az} & to & V_{bz} \end{Bmatrix}$$

$$AccelerationRange = \begin{Bmatrix} a_{ax} & to & a_{bx} \\ a_{ay} & to & a_{by} \\ a_{az} & to & a_{bz} \end{Bmatrix}$$

Using the above speed and acceleration range as a boundary condition of Quintic Polynomial formula and can find the values of the variable defined in the formula.

Let’s consider the starting and end position variables like position, velocity, and acceleration for each direction.

For X direction,

$x(t_0') = x_i$

$x(t_f^{'}) = x_{i+1}$

$x(\dot{t}_0^{'}) = v_{ax}$

$x(\dot{t}_0^{'}) = v_{bx}$

$x(\ddot{t}_0^{'}) = a_{ax}$

$x(\ddot{t}_0^{'}) = a_{bx}$

For Y direction,

$y(t_0^{'}) = y_i$

$y(t_f^{'}) = y_{i+1}$

$y(\dot{t}_0^{'}) = v_{ay}$

$y(\dot{t}_0^{'}) = v_{by}$

$y(\ddot{t}_0^{'}) = a_{ay}$

$y(\ddot{t}_0^{'}) = a_{by}$

For Z direction,

$z(t_0^{'}) = y_i$

$z(t_f^{'}) = y_{i+1}$

$z(\dot{t}_0^{'}) = v_{az}$

$z(\dot{t}_0^{'}) = v_{bz}$

$z(\ddot{t}_0^{'}) = a_{az}$

$z(\ddot{t}_0^{'}) = a_{bz}$

According to the above boundary conditions, positions of the end effector is given by,

$$p_x = \begin{bmatrix} 1 & t_j & t_j^2 & t_j^3 & t_j^4 & t_j^5 \\ 0 & 1 & 2t_j & 3t_j^2 & 4t_j^3 & 5t_j^4 \\ 0 & 0 & 2 & 6t_j & 12t_j^2 & 20t_j^3 \\ 1 & t_{j+1} & t_{j+1}^2 & t_{j+1}^3 & t_{j+1}^4 & t_{j+1}^5 \\ 0 & 1 & 2t_{j+1} & 3t_{j+1}^2 & 4t_{j+1}^3 & 5t_{j+1}^4 \\ 0 & 0 & 2 & 6t_{j+1} & 12t_{j+1}^2 & 20t_{j+1}^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}$$

$$p_y = \begin{bmatrix} 1 & t_j & t_j^2 & t_j^3 & t_j^4 & t_j^5 \\ 0 & 1 & 2t_j & 3t_j^2 & 4t_j^3 & 5t_j^4 \\ 0 & 0 & 2 & 6t_j & 12t_j^2 & 20t_j^3 \\ 1 & t_{j+1} & t_{j+1}^2 & t_{j+1}^3 & t_{j+1}^4 & t_{j+1}^5 \\ 0 & 1 & 2t_{j+1} & 3t_{j+1}^2 & 4t_{j+1}^3 & 5t_{j+1}^4 \\ 0 & 0 & 2 & 6t_{j+1} & 12t_{j+1}^2 & 20t_{j+1}^3 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

$$p_x = \begin{bmatrix} 1 & t_j & t_j^2 & t_j^3 & t_j^4 & t_j^5 \\ 0 & 1 & 2t_j & 3t_j^2 & 4t_j^3 & 5t_j^4 \\ 0 & 0 & 2 & 6t_j & 12t_j^2 & 20t_j^3 \\ 1 & t_{j+1} & t_{j+1}^2 & t_{j+1}^3 & t_{j+1}^4 & t_{j+1}^5 \\ 0 & 1 & 2t_{j+1} & 3t_{j+1}^2 & 4t_{j+1}^3 & 5t_{j+1}^4 \\ 0 & 0 & 2 & 6t_{j+1} & 12t_{j+1}^2 & 20t_{j+1}^3 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix}$$

According to the system, Initial point and final points positions,velocities, acceleration conditions given by,

$$b = [x_0; \dot{x}_0; \ddot{x}_0; x_0; \dot{x}_f; \ddot{x}_f] \tag{4.7}$$

Based on the Equation 4.7, we can define the initial, middle, and final positions, velocities, and accelerations as given in Equations no 4.8 ,4.9 ,4.10.(first and last points have always zero speed)[MATLAB Code:Appendix B]

For the first position,

$$b = [x_{j.k}; 0; 0; x_{j+1,k}; V_{j+1,k}; 0] \tag{4.8}$$

For the middle position

$$b = [x_{j.k}; V_{j.k}; 0; x_{j+1,k}; V_{j+1,k}; 0] \tag{4.9}$$

For the last position

$$b = [x_{j.k}; V_{j.k}; 0; x_{j+1,k}; 0; 0] \tag{4.10}$$

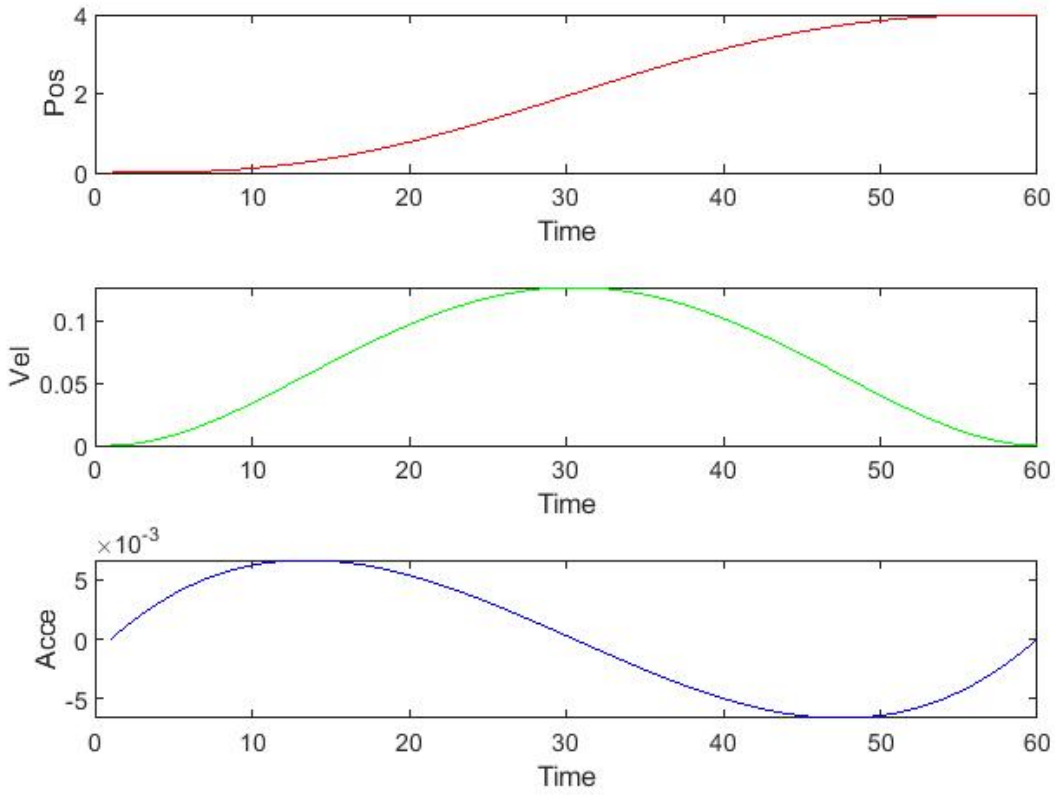

Figure 4.7: 5th Order Polynomial Position, Velocity and Acceleration Diagram

Considering above graphs, the robot end effector has not any acceleration in start point and end point.

Robot X direction speed = v_x

Robot y direction speed = v_y

Robot z direction speed = v_z

Consider the absolute speed for,

$$V_x^{abs} = \overrightarrow{v_x} - \overleftarrow{v_c}$$

$$V_x^{abs} = \overrightarrow{v_x} + \overrightarrow{v_c}$$

$$V_x^{abs} = \overrightarrow{v_x + v_c} \tag{4.11}$$

$$V_y^{abs} = v_y \tag{4.12}$$

$$V_z^{abs} = v_z \tag{4.13}$$

Consider primarily the point-to-point movement while creating a trajectory. The starting and endpoint variables should be determined using absolute velocity and acceleration according to the point trajectory. The absolute velocity can be calculated using the robot's movement vectors, as indicated in Equations Nos. 4.11,4.12 and 4.13.

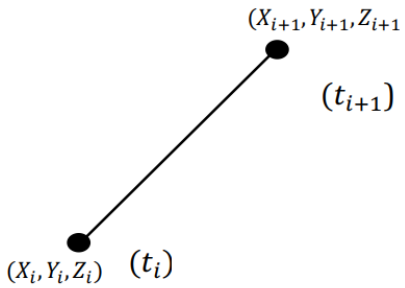

Figure 4.8: End-effector start point and end point coordinates

According to the above figure the starting point coordinate is (X_i, Y_i, Z_i) and endpoint is $(X_{i+1}, Y_{i+1}, Z_{i+1})$. using absolute velocities, the endpoint coordinate can be defined as following equation no 4.14,4.15 and 4.16 with respect to the starting position.

End effector movement for X direction,

$$X_{i+1} = X_i + V_x^{abs}(t_{i+1} - t_i) + \frac{1}{2}a_x(t_{i+1} - t_i)^2$$

$$X_{i+1} = X_i + (v_x + v_c)(t_{i+1} - t_i) + \frac{1}{2}a_x(t_{i+1} - t_i)^2 \tag{4.14}$$

End effector movement for Y direction,

$$Y_{i+1} = Y_i + V_y^{abs}(t_{i+1} - t_i) + \frac{1}{2}a_y(t_{i+1} - t_i)^2$$
$$Y_{i+1} = Y_i + v_y(t_{i+1} - t_i) + \frac{1}{2}a_y(t_{i+1} - t_i)^2 \quad (4.15)$$

End effector movement for Z direction,

$$Z_{i+1} = Z_i + V_z^{abs}(t_{i+1} - t_i) + \frac{1}{2}a_z(t_{i+1} - t_i)^2$$
$$Z_{i+1} = Z_i + v_z(t_{i+1} - t_i) + \frac{1}{2}a_z(t_{i+1} - t_i)^2 \quad (4.16)$$

4.3 Conveyor Speed Controlling System

The DC motor is an electric actuator that is highly controllable. DC motors are used extensively in industrial control applications. In speed control aspects, DC motors are extremely versatile and flexible. High-performance DC motor drives are common in industrial applications such as conveyors, autonomous vehicles, robotic manipulators, etc.DC motors are considered adjustable velocity actuators. That means the DC motor drive is best suited for variable speed applications, including accurate speed and torque. The DC motor control system was used in this study to change the conveyor speed and adjust the speed while accounting for external torque or disturbances.

The DC motor system utilizes three common speed control techniques. Control of Field Resistance, Control of Armature Resistance, and Control of Armature Voltage are the three.The speed of a separately excited DC motor is directly proportional to the motor's armature voltage. PI or PID controllers are used for speed control under varying load conditions or disturbances of the system

and control the armature voltage according to the speed. Under this approach, the PID controller is used to overcome the unwanted undershoot of motor speed due to changing load impact in certain conditions. The MATLAB Simulink platform is used to model the PID controller for that proposed system and simulate that system under certain conditions. Figure 4.9 shows the closed-loop DC motor controlling system for the proposed conveyor speed control system. [13].

The DC motor model is developed by MATLAB Simulink and simulated under load conditions.

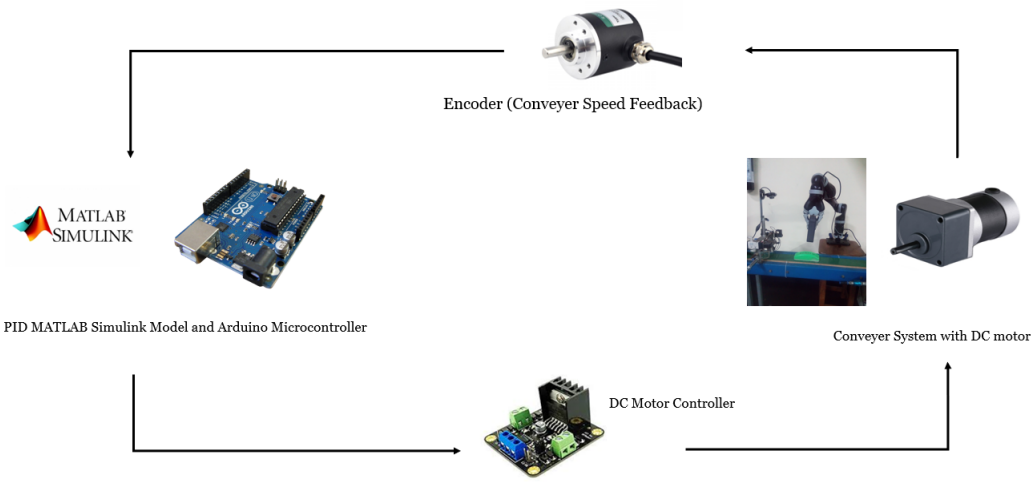

Figure 4.9: Conveyor Speed controller System

4.3.1 Conveyor Motor System Model

The main assumption of that system is that the friction between the object to be scanned and the conveyer belt surface is sufficient to maintain the grip and fixed orientation from the start to the end. And also, the belt will not slip on the pulley. Conveyer system parameters can be defined as follows,

Driver Pully Diameter/Motor (ϕ_m)

Driven Pully Diameter/Conveyer (ϕ_c)

Conveyer Speed (V)

Target object weight (Mg)

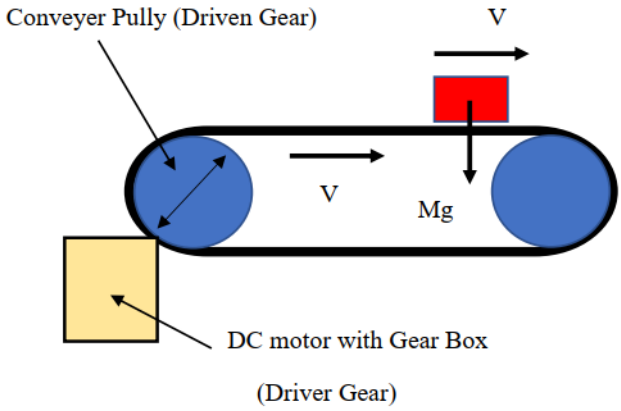

Figure 4.10: Conveyer Model with DC gear Motor

Conveyer pully Speed (ω_p)

Motor Shaft speed (ω_m) Using above variables conveyer speed can be define as,

$$V = \frac{\phi_c}{2} \times \omega_p \tag{4.17}$$

But consider the speed ratio of the motor and diameter of the pulleys,

$$\frac{\phi_p}{\phi_m} = \frac{\omega_m}{\omega_p} \tag{4.18}$$

Consider equation 4.17 and 4.18

$$V = \frac{\phi_c}{2} \times \omega_m \times \frac{\phi_m}{\phi_p} \tag{4.19}$$

Analytical velocity of the conveyer system is given in equation no 4.19. Then we measure the conveyer speed using an encoder. $\omega_m^{'}$ is the measured velocity of the conveyer system. And ω_m is the speed set point of the conveyer. Then the error can be calculated as follows,

$$Error(e) = \omega_m^{'} - \omega_m \tag{4.20}$$

Then, based on the mathematical function of the system, we can model the control system with a PID controller.

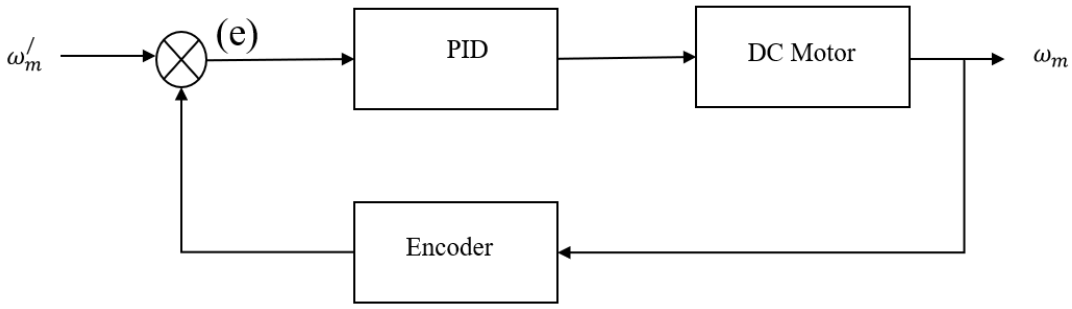

Figure 4.11: Closed Loop Feedback System for DC motor

4.3.2 PID Controller

PID Controller system can be define as,

$$PID = K_P \times e_\omega + K_I \times \int e_\omega dt + K_D \times \frac{de_\omega}{dt}$$

but in our controller, we use to mathematical method to calculate Integral and Derivative part.

Then,

$$PID = K_P(\omega_m^{'} - \omega_{mt}) + K_I(\sum_{t=0}^{t} e_{\omega t}) + K_D(\frac{e_{\omega t} - e_{\omega t+1}}{\Delta t}) \tag{4.21}$$

4.3.3 DC Motor Simulink Model

The closed-loop system is used to model the armature-controlled DC motor speed controller system. The MATLAB Simulink software is used to simulate the armature control DC motor [14]. In this system, the speed is controlled by the applied armature voltage and the field current. Then in this case mainly focusing

about motor shaft speed and armature voltage relationship. Therefore,

Relationship between motor speed Vs applied armature voltage transfer function gives as equation no 4.22.

$$\frac{\omega_{(s)}}{V_{a(s)}} = \frac{K_m}{L_a.J_m.S^2 + (R_a.J_m + L_a.B_m)S + (R_a.B_m + K_bK_m)} \tag{4.22}$$

According to the motor constant and other parameters, the closed-loop transfer function can be calculated as follows. [MATLAB code:Appendix D]

```
Closed loop TF of DC motor with PID controller

ans =

          0.1 s^2 + 1.4 s + 3.4
  ---------------------------------
  0.008 s^3 + 0.22 s^2 + 1.8 s + 3.4

Continuous-time transfer function.
```

Figure 4.12: Closed Loop TF of DC motor PID controller

The Simulink DC motor model for this above system as shown in Figure 4.13,

Then run the MATLAB Simulink model and change the motor constant and variables in MATLAB script the results of the Motor can plot as shown in figure 4.14.

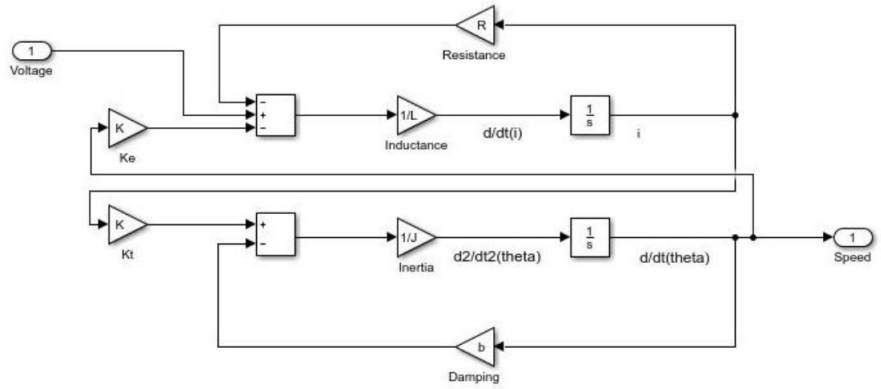

Figure 4.13: DC motor MATLAB Simulink Model

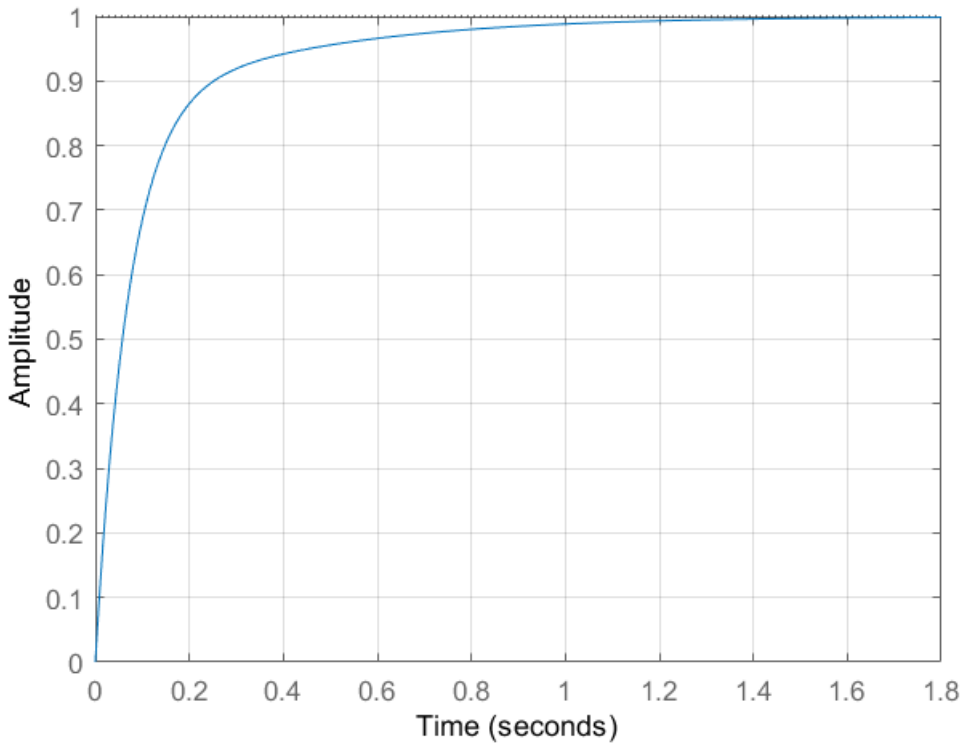

Figure 4.14: PID Controller Results for Conveyer DC Motor.

Chapter 5

RESULTS AND DISCUSSION

The movement of the robot manipulator and the object orientation of the moving object are the main tasks of this study. We take the system output for path planning of the moving object using the newly created mechanism. Our system outputs can be described as follows, based on our study objectives.

5.1 Simulation Results for Robot Manipulator Configuration

The 4 DoF manipulator kinematics simulation was done by using the Robot analyzer toolbox in MATLAB. It is used to analyze the robot manipulator Link configuration, end effector configuration, and also end-effector point coordinates and joint parameters.

5.1.1 Link Configuration

According to the DH parameters, the model of a 4 DoF manipulator and transformation metrix for each link can be generated through this toolbox. First, the link parameters need to import this toolbox and take the configurations according to our requirements. Figure 5.1 shows the link configuration simulation results of the MATLAB robotics toolbox. As an example, the first link's link configuration matrix from the base frame is shown below.

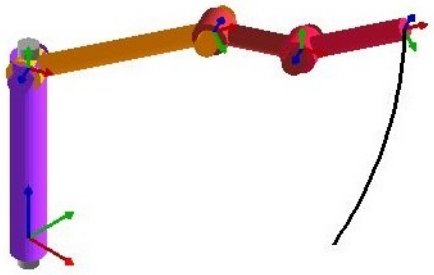

Figure 5.1: Link Configuration of 4Dof Robot Manipulator

$$\text{Link configuration matrix for first link} = \begin{bmatrix} 1 & 0 & 0 & 0.2755 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0.4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5.1.2 End Effector Configuration

According to the DH parameters, the end effector transformation metrix can be generated by using a 4 DoF robot arrangement as shown in Figure 5.1. Using the End effector configuration metrix, the position of the end effector can be simulated on the MATLAB simulink platform.

$$\text{End effector configuration matrix} = \begin{bmatrix} 0.818 & 0.385 & -0.426 & 0.479 \\ 0.385 & 0.181 & 0.904 & 0.226 \\ 0.426 & -0.904 & 0 & 0.467 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5.1.3 End Effector Point Coordinates and Joint Parameter

Using a robot manipulator model, the behaviors of the end-effector can be simulated according to the positions, velocities, and accelerations of each joint.

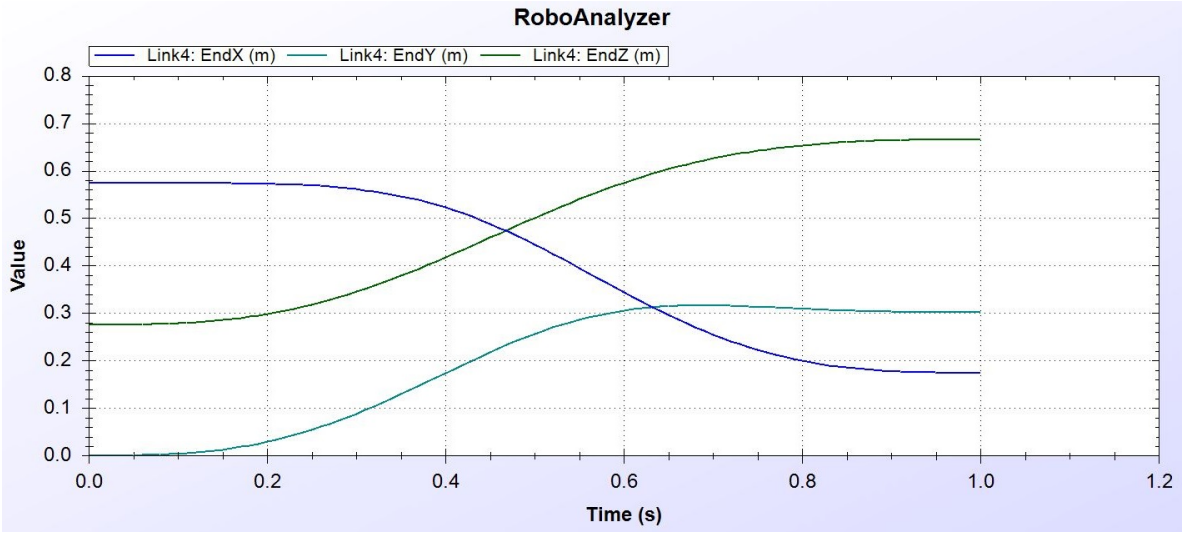

Figure 5.2: End effector Point Coordinate Graph

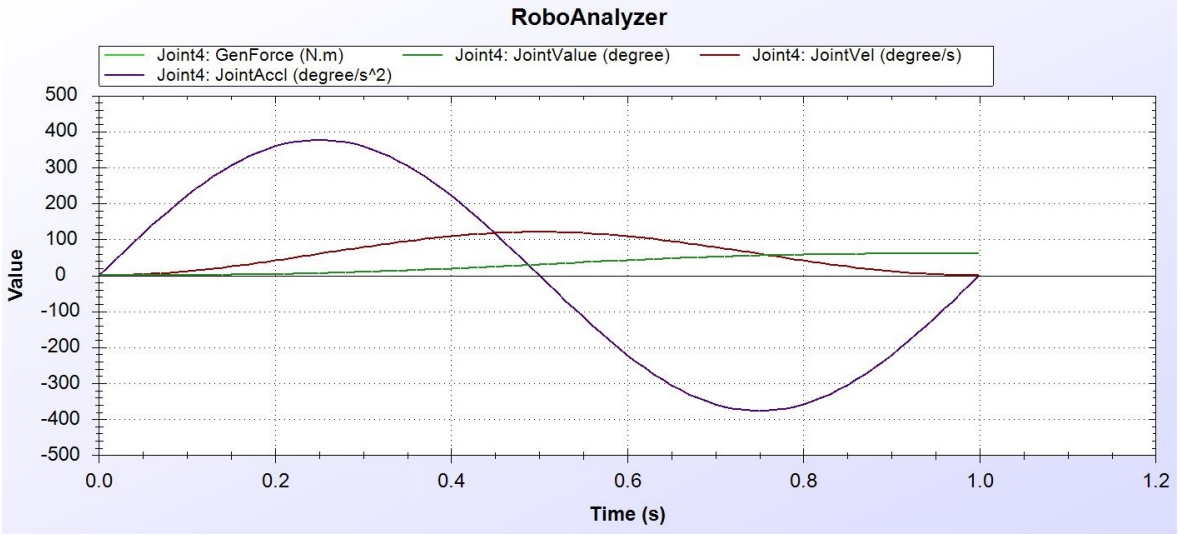

Figure 5.3: Manipulator Joint parameters

In this case, it mainly considers the end-effector point coordinates and joint parameters.Design of the 4 DoF robot manipulator system on the Robot Analyzer toolbox based on the DH parameter. The simulation model is shown in Figure no 5.1.

Simulation results of the end effector point coordinates are shown in Figure 5.2. Also, manipulator joint parameters results of the system are shown in Figure 5.3.In this graph we can identify the joint parameters like force, joint angle, joint velocity and acceleration for the 4 DOF robot joints. According to this result, we can identify the end-effector joint parameter for simulating the quentic polynomial formula.

5.2 Robot Manipulator Kinematic Simulation

The most commonly used robotic toolbox is the "Peter Corke" MATLAB Robotic toolbox for simulating any type of robot system in the MATLAB workspace. Therefore, this toolbox is used to simulate the kinematics of the robot manipulator. We can include the manipulator link parameters and joint parameters into the system and it can be modeled as a 3D Robot system on a graphical user interface (GUI). Then, using MATLAB workspace, we can change the point coordinates and joint coordinates to analyze their forward kinematics and Inverse Kinematics.

5.3 Forward Kinematics

The Forward Kinematics model for the 4 DoF Robot manipulator has already been developed based on the DH table. This toolbox is used to simulate the 4DoF robot manipulator and also develop the GUI for this purpose and is shown in Figure 5.4.

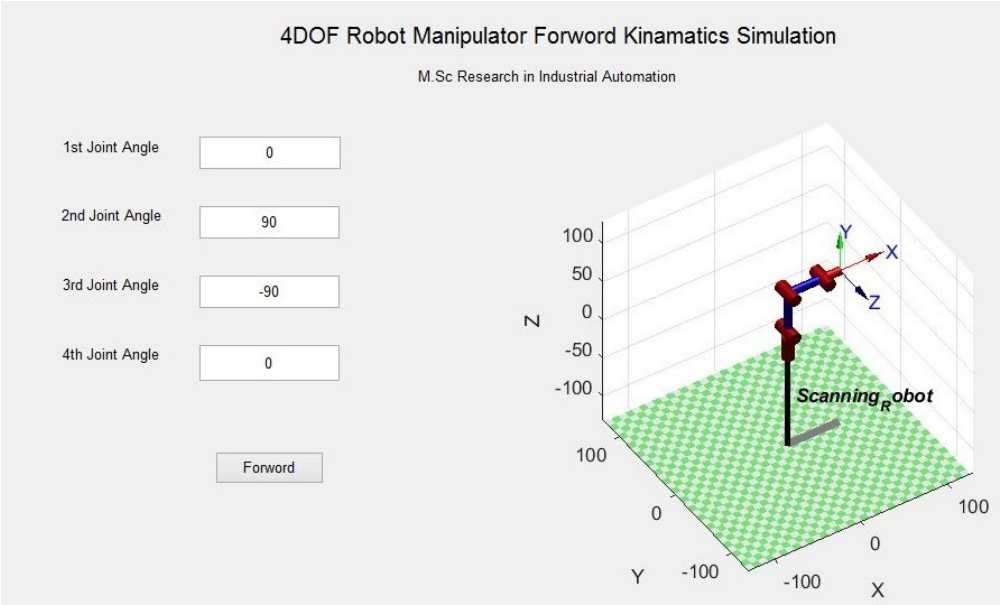

Figure 5.4: 4DoF Robot Manipulator Forward Kinematic Simulation

5.4 Inverse Kinematics

Figure 5.5 shows the Inverse Kinematics model simulation GUI of the 4DoF robot manipulator which is used in the scanning process. It helps to identify and simulate the waypoints which are generated in the CAD model. According to the point coordinates of the way-point, we can simulate the path trajectory based on this GUI.It is used to verify the way points which are generated from the manual scanning process.

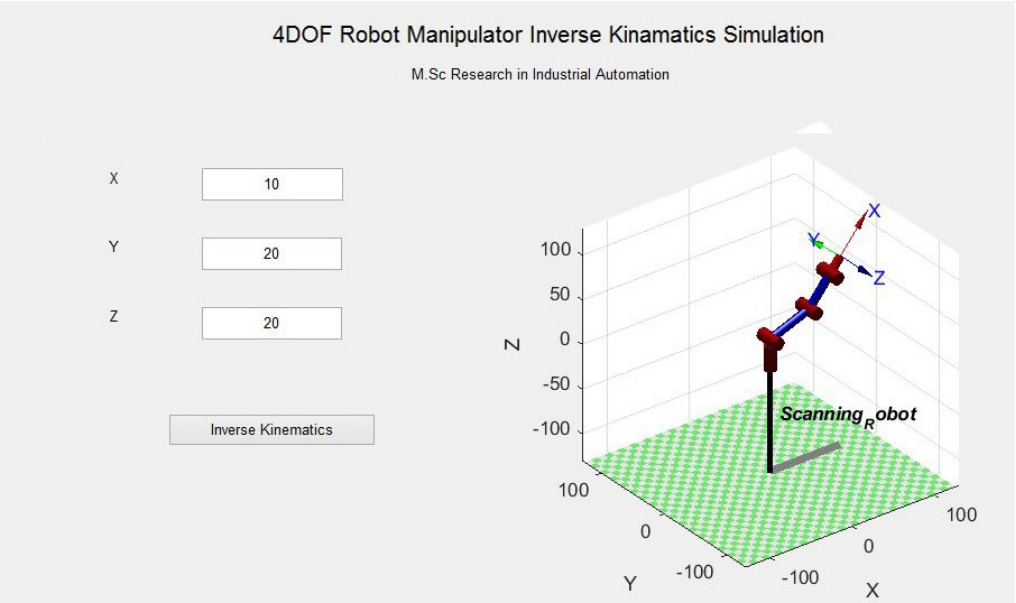

Figure 5.5: 4 DoF Robot Manipulator Inverse Kinematics Simulation

5.5 Simulation Results for Optimum Conveyor Speed

The path trajectory was changed according to the object shape. Therefore, the robot's end effector movement and speeds need to change according to the robot parameters. Therefore, we use a mathematical equation to calculate the end effector's moving speed. The X direction movement speed is equal to conveyor speed due to minimizing the relative speed.

Using the equations and values mentioned in chapter 4.2.1, simulate the proposed system using MATLAB. Appendix C contains the complete MATLAB code.In that simulation, we need to check the optimum speed of the conveyor by changing the different speed values. In this case, we checked the conveyor speed at $0ms^{-1}$,$0.1ms^{-1}$ and $0.4ms^{-1}$ respectively. When increasing the convayer speed, the end-effctor X direction speed also increases. As a result of that X-direction speed, it generates some jerks when increasing the speed. The simulation results are also shown in Figures 5.7,5.8 and 5.9.In the graph, the red color line indicates the X direction position of the end effector. The other green and blue colored

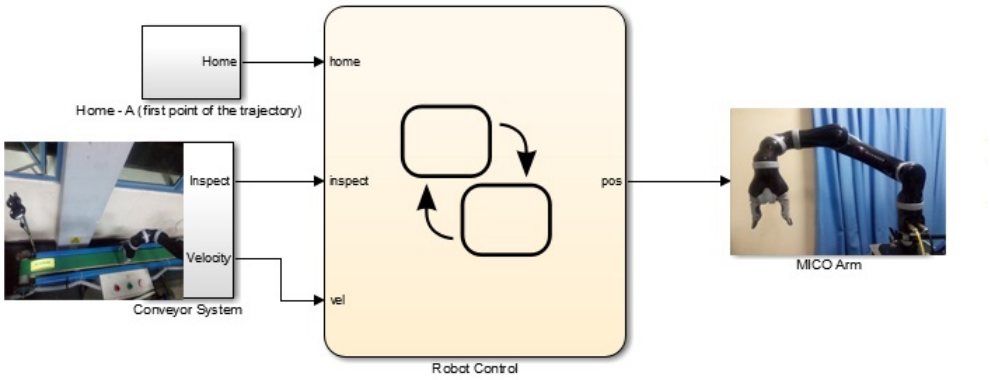

Figure 5.6: MATLAB Simulink model for Robot Trajectory Planning

graph lines indicate the Z position and X position of the end effector respectively. According to the manufacturer's requirements, the robot's end effector's maximum linear speed is equal to $0.2ms^{-1}$. Therefore, we can check the optimum speed based on the graph. Therefore, we can recommend that conveyer speed be maintained between $0ms^{-1}$ to $0.2ms^{-1}$. We take the mean value and simulated value of the conveyor speed is $0.1ms^{-1}$ as the optimum conveyor speed of the system.

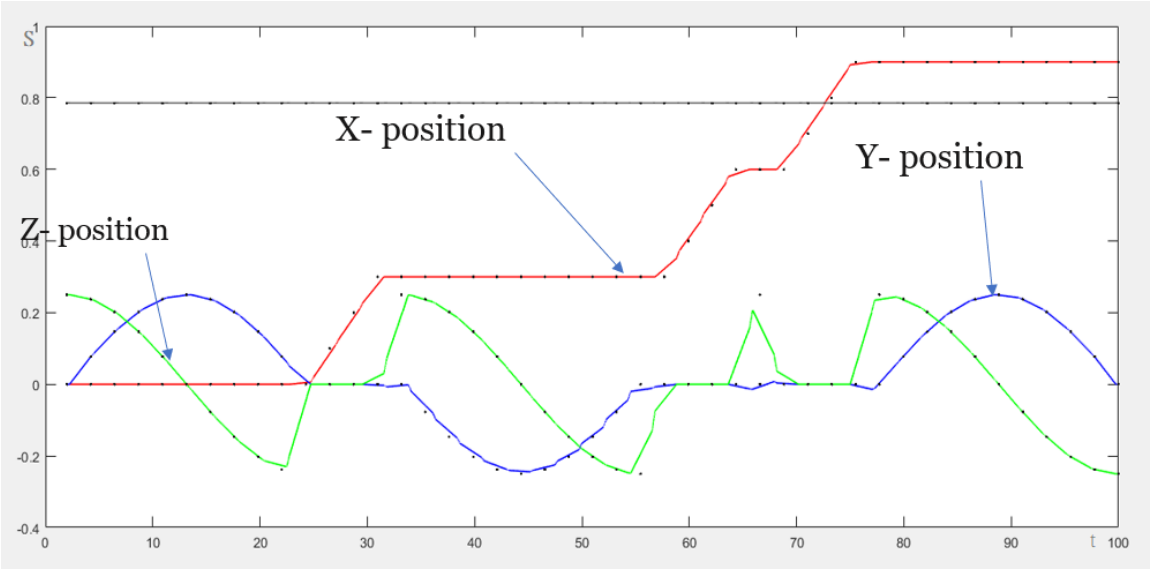

Figure 5.7: Conveyer Speed Vc=0

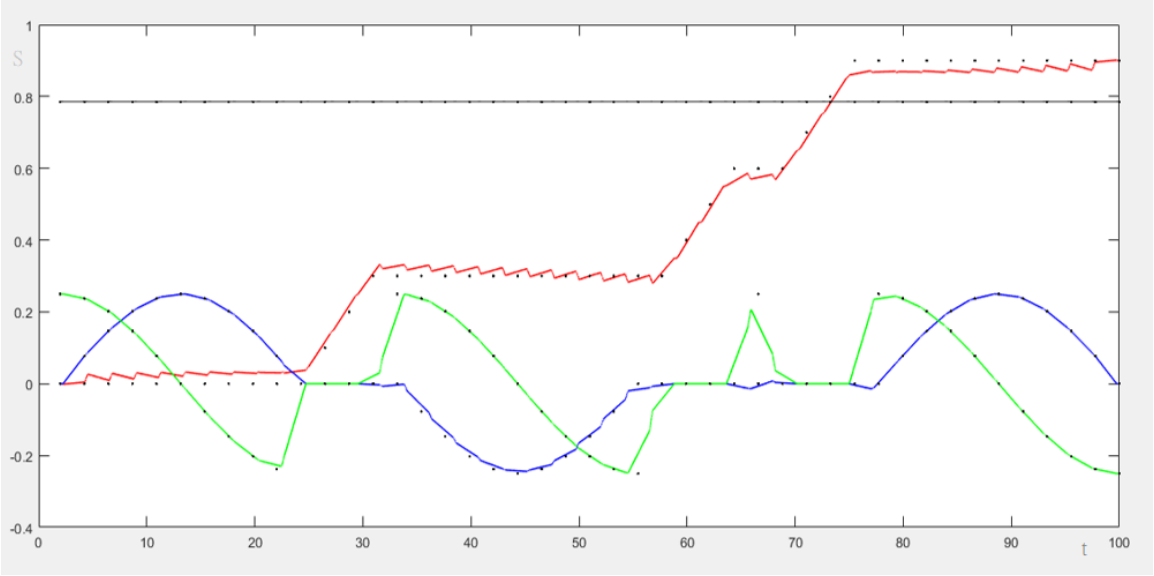

Figure 5.8: Conveyer Speed Vc=0.1 m/s

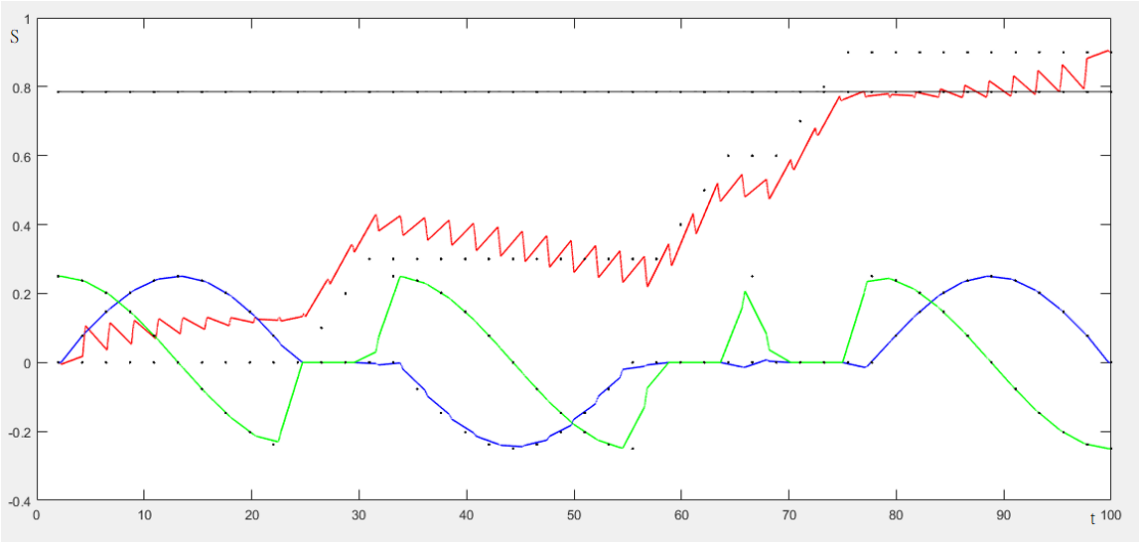

Figure 5.9: Conveyer Speed Vc=0.4 m/s

5.6 Object Orientation Detection Validation Results.

The validation method is based on the actual angle, and measured angle data sets of the above mentioned method in chapter 3.6. Considering the known angle value of that object orientation and measuring the software base angle value to analyze the error difference between the actual angle and the measured angle. According to the error difference, we can identify the accuracy of the proposed method for identifying the objects' orientation correctly.

The MATLAB software platform is used to simulate the object orientation detection system. The webcam module is used to capture the image of objects moving on a conveyor. Then it saves it to the MATLAB folder and starts to analyze the angle according to our reference image. After calculating the image angle, its value is fed into the Simulink model of the robot system.

According to that process, analyzing time is important to consider with respect to all system operation times. Therefore, we measure the image analyzing time by using a special MATLAB command [Appendix E]. To analyze the orientation detection process speed, there are two types of computers used to run

the MATLAB ".m" file, and the results of the angle and processing time can be analyzed.

The analysis of data sets for four different objects using two different computers is shown below. The Figures 5.10 and 5.11 show the object 01 (Duster surface) feature extraction process and the object 02 (contactor relay surface) feature extraction process respectively. Figure 5.12 and Figure 5.13 also use the result feature extraction process for another two objects.

Tables 5.1, 5.2, 5.3, and 5.4 show the object's actual angle, measured angle, accuracy, and feature extraction processing time using two different computations. We analyze the angle measurement variation for the first two objects and plot the graph based on the number of positions with respect to the object's angle.According to the graph, there is no significant difference between the actual and measured values of the first two objects.

Figure 5.10: Object 01(Contactor relay plastic cover) surface feature extraction in different angular positions

The difference between the actual angle and the measured angle of Table No 5.1 is less than 5^0.The maximum deference is 4^0 and it affects the large angle value measurement of 255^0.but other angle measurements are recorded as minimum error deference.Considering Table No 5.2, there is no large angle difference between the actual and measured angle. Its highest difference angle value is 1.8^0.Therefore, object no 01 and object no 02 angle measurements are more accurate for our feature extraction method.

Table 5.1: Results of variation of angle value with respect to actual angle value and processing time for object no 01 (Contactor relay plastic cover).

Act: angle(Deg:)	Meas:angle	Accuracy	Processing Time(S)(PC 01)	Processing Time(S)(PC 02)
0	0.1	+0.1	9.672	2.170
45	44.9	-0.1	9.296	2.237
90	90	0	9.281	2.190
135	134.5	-0.5	9.460	2.120
180	179.7	-0.3	9.442	2.063
255	229	+4.0	9.461	2.178
270	272.1	+2.1	9.362	2.104
315	315.3	+0.3	9.306	2.187

Table 5.2: Results of Variation of angle value with respect to actual angle value and Processing time for object no 02 (Duster plastic cover).

Act: angle(Deg:)	Meas:angle	Accuracy	Processing Time(S)(PC 01)	Processing Time(S)(PC 02)
0	0.2	+0.2	9.983	2.356
45	44.9	-0.1	10.563	2.476
90	90.1	+0.1	9.818	2.365
135	134.5	-0.5	9.901	2.280
180	179.7	-0.3	9.674	2.274
255	225.2	+0.2	9.871	2.280
270	270.1	+0.1	9.833	2.299
315	313.2	-1.8	9.979	2.426

Table 5.3: Results of Variation of angle value with respect to actual angle value and Processing time for object no 03 (Switch cover).

Act: angle(Deg:)	Meas:angle	Accuracy	Processing Time(S)(PC 01)	Processing Time(S)(PC 02)
0	0.3	+0.3	10.672	3.271
45	40.9	-4.1	9.236	2.137
90	94	+4.0	9.381	2.391
135	155.5	+20.5	9.260	2.250
180	185.7	+5.7	9.412	2.363
255	229	-26.0	9.562	2.572
270	273.1	+3.1	10.332	2.344
315	335.3	+20.3	9.502	3.187

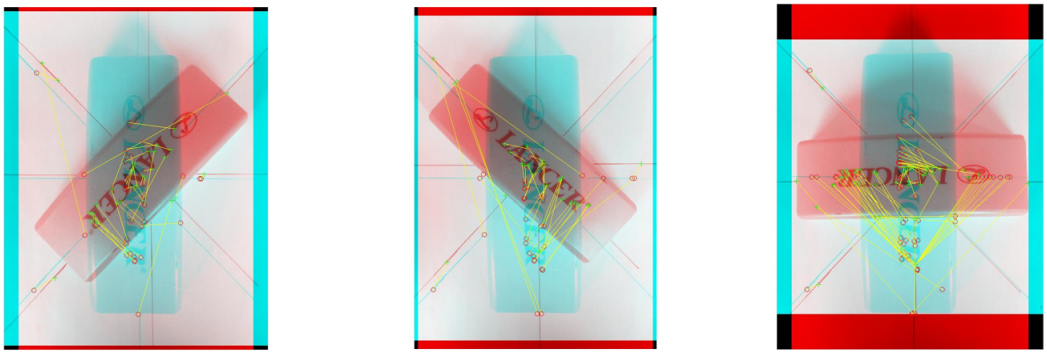

Figure 5.11: Object 02 (Duster) Surface feature Extraction in different angular positions

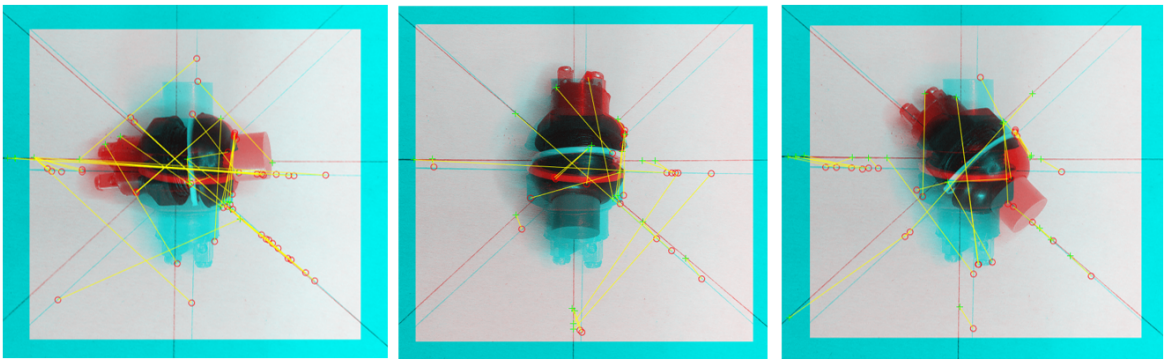

Figure 5.12: Object 03 (Switch Cover) Surface feature Extraction in different angular positions

There is the highest variation in the measured angle accuracy for the 3rd and 4th objects. The maximum angle difference between the 3rd object is 26.0^0 and the 4th object's maximum angle difference is 34.5^0. Compared with other recorded values of each table, it is too large a difference. It may happen because of the complexity of the object surface. According to our research limitations, we mention the system for using the object without any high-complexity surface. Therefore, the analysis of the results part is considered by using the first two objects (object no 01 and object no 02).

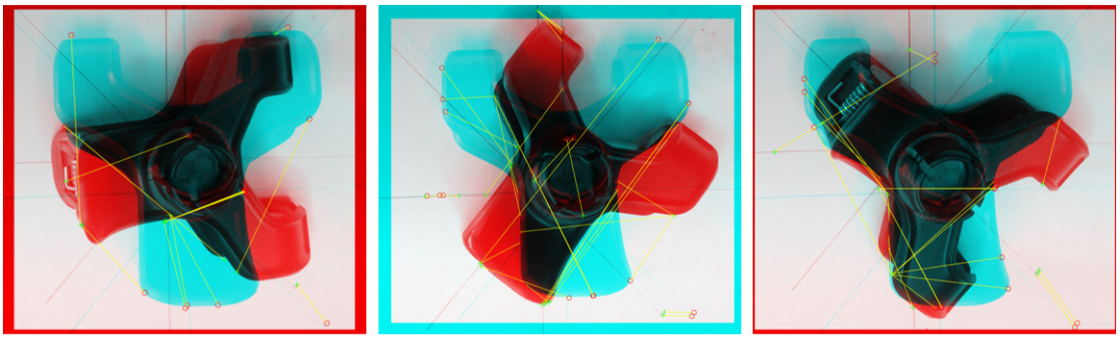

Figure 5.13: Object 04 (Plastic Gripper) Surface feature Extraction in different angular positions

Table 5.4: Results of Variation of angle value with respect to actual angle value and Processing time for object no 04 (Plastic gripper).

Act: angle(Deg:)	Meas:angle	Accuracy	Processing Time(S)(PC 01)	Processing Time(S)(PC 02)
0	0.5	+0.5	10.672	2.070
45	42.9	-2.1	11.296	2.213
90	93.5	+3.5	10.231	2.340
135	100.5	-34.5	9.260	2.250
180	189.7	+9.7	8.452	2.162
255	230	-25.0	9.562	2.368
270	275.2	+5.2	9.952	2.504
315	345.3	+30.3	9.831	2.485

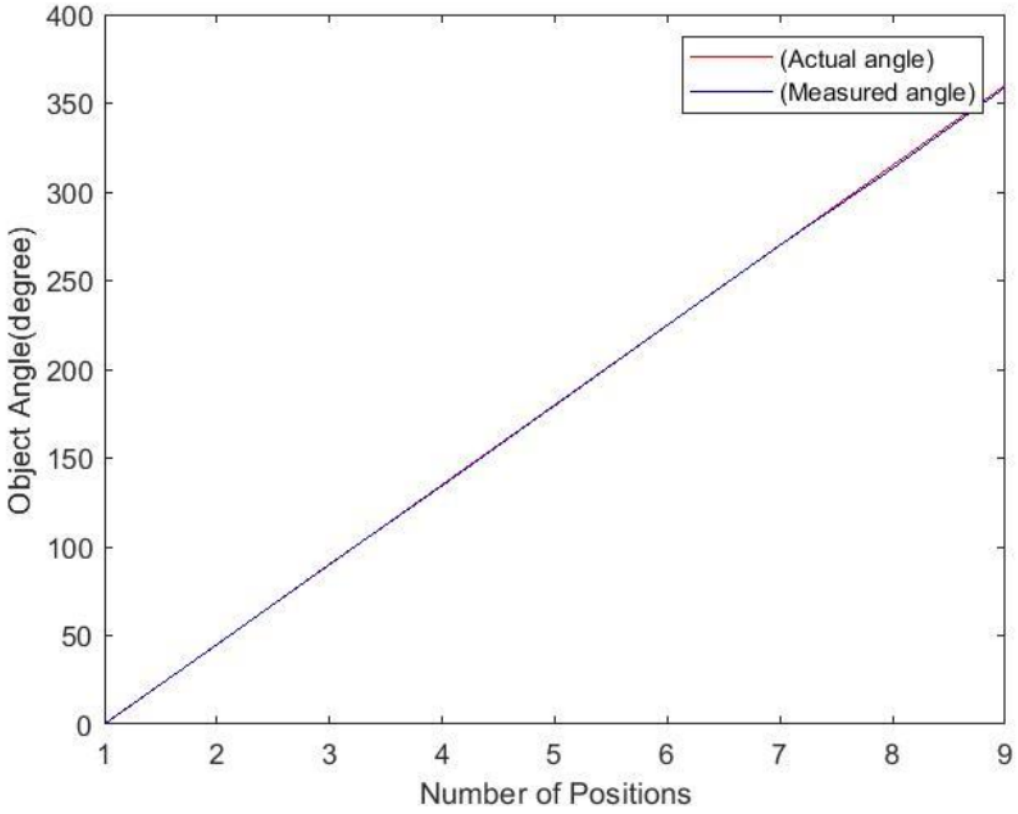

Figure 5.14: Graph of Actual angle value and Measured value variation for object 01(Contactor relay cover)

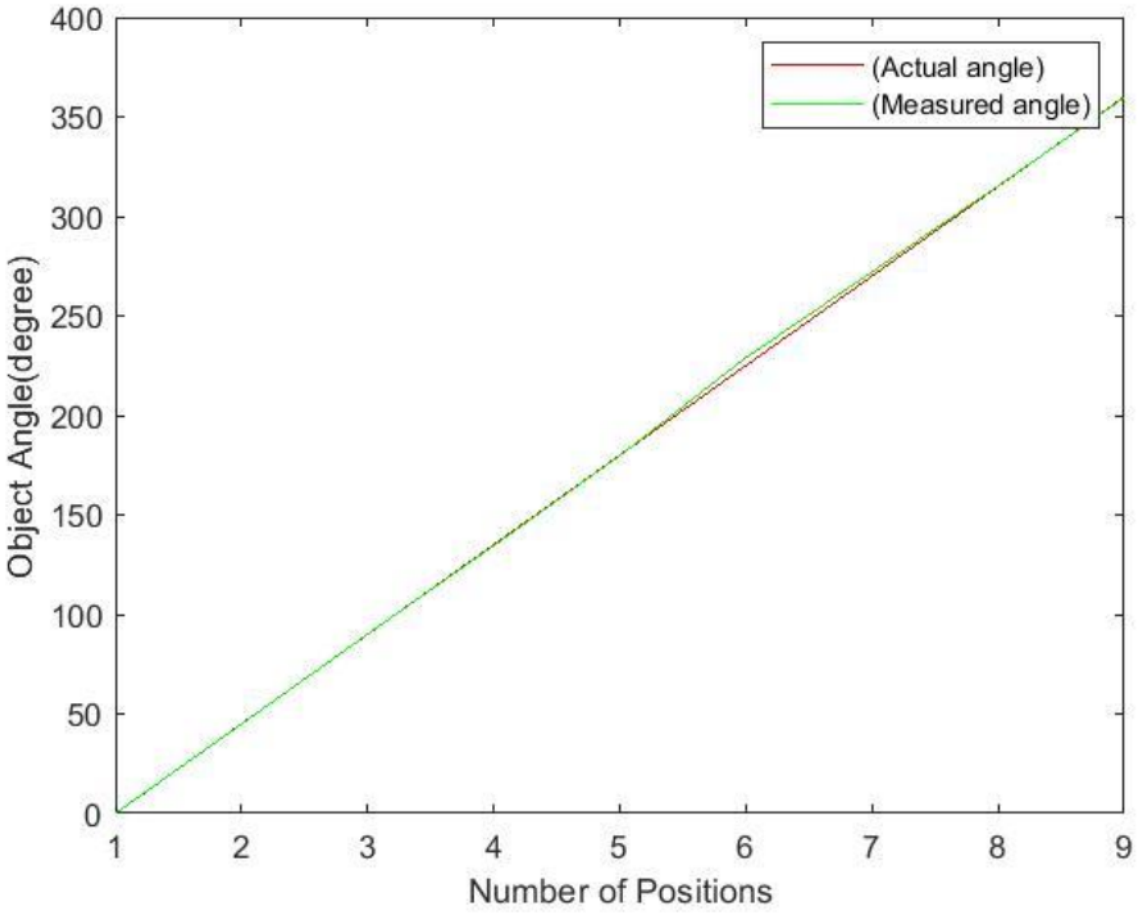

Figure 5.15: Graph of Actual angle value and Measured value variation for Object no 02(Duster cover)

Chapter 6

CONCLUSION

A 3D Sense scanner is used to scan a moving object on a conveyor and generate a 3D model by maintaining the distance and the speed of the scanner. The generated 3D model is imported into the CAD software and divided into the same slices. Using the slice, find the point coordinates of the path from the generated profile. Boundary conditions are included in the profiles and generate the point trajectory. According to the path shape and point trajectory, it generated the path trajectory for the scanned 3D model. The generated trajectory is used to move the robot manipulator end effector with the scanner to get high quality scanning output.

The moving speed changes according to the object's shape and the conveyor's conditions. Therefore, optimum speed needs to be calculated. The end effector speed is calculated using mathematical equations according to the conveyor speed. The optimum end effector speed is set at $0.1ms^{-1}$.According to the manufacturer's specifications, the maximum end effector linear speed is $0.2ms^{-1}$.Therefore, the calculated optimum speed is the most suitable for the scanning operation. The calculated speed is fed as the set speed to the conveyor motor PID controller system. According to the conveyor speed, the end effector speed is mapped to the conveyor speed to minimize the relative speed in the X direction.

Object orientation is measured to change the scanner orientation, which is perpendicular to the target object. The object surface feature extraction method

is used to measure the orientation angle of each object with respect to the object reference image. The distance between the scanner and the target object is measured by using an ultrasonic sensor. The distance value is used to adjust the waypoint regarding the generated trajectory to get an accurate reading for object quality scanning.

The robot manipulator system is simulated using the MATLAB robotic toolbox and the "Peter Corke" Robot toolbox. Analyze Forward Kinematics, Inverse Kinematics, and point to point trajectory planning by using these robots' toolbox.It helps to identify the path waypoints before implementing the end-effector path planning for the scanning process.

The proposed method can be used for object surface scanning or surface feature identification of moving objects on a conveyor. The scanner device is attached to a robot manipulator and the robot end-effector is moved based on the path which is generated using this proposed method. These types of methods are very useful for the injection molding industry to identify their product quality and surface finishing.

Future steps include performing object defect analysis and rejecting damaged and other objects from the conveyor. Also, try to implement various object identification methods and implement this system to select the suitable object and automatically generate a path trajectory without any research limitations.

LIST OF PUBLICATIONS

- Weerasekara D.R.B.K.K, Kumara K. J. C., "Motion Planning of a Robotic Manipulator for 3D Scanning of Moving Object on a Convayor", 16^{th} Academic Session of University of Ruhuna ,Matara,Sri Lanka,6th March 2019.

BIBLIOGRAPHY

[1] S. S. Pawar, H. Mithaiwala, A. Gupta, and S. Jain, "Review paper on design of 3d scanner," in *2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, 2017, pp. 650–652.

[2] R. ZHAO, "Trajectory planning and control for robot manipulations," Theses, Université Paul Sabatier - Toulouse III, Sep. 2015. [Online]. Available: https://tel.archives-ouvertes.fr/tel-01285383

[3] Y. Luan, W. Xu, J. Li, D. Zhou, H. Wang, and H. Ji, "Kinematics modeling and simulation of a 4-dof manipulator," in *2017 International Conference on Computer Systems, Electronics and Control (ICCSEC)*, 2017, pp. 302–305.

[4] A. Menon, B. Cohen, and M. Likhachev, "Motion planning for smooth pickup of moving objects," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 453–460.

[5] J. Gyorfi and C.-H. Wu, "A minimum-jerk speed-planning algorithm for coordinated planning and control of automated assembly manufacturing," *IEEE Transactions on Automation Science and Engineering*, vol. 3, no. 4, pp. 454–462, 2006.

[6] A. Cowley, B. Cohen, W. Marshall, C. J. Taylor, and M. Likhachev, "Perception and motion planning for pick-and-place of dynamic objects," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 816–823.

[7] R. Venkatesan and A. B. Ganesh, "Real time implementation on moving object tracking and recognisation using matlab," in *2012 International Conference on Computing, Communication and Applications*, 2012, pp. 1–8.

[8] I. I. Lychkov, A. N. Alfimtsev, and S. A. Sakulin, "Tracking of moving objects with regeneration of object feature points," in *2018 Global Smart Industry Conference (GloSIC)*, 2018, pp. 1–6.

[9] S. Rianmora, K. Nuamchit, N. Vanasrivilai, P. Tantipiched, and A. Rammbuth, "Applying scanning techniques to create 3d model," in *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, 2015, pp. 005 217–005 222.

[10] *3D Systems Sense User Manual*, 3rd ed., 3D Systems,Inc., 333 Three D Systems Circle,Rock Hill,SC,29730, 2013.

[11] J. Xiao, W. Han, and A. Wang, "Simulation research of a six degrees of freedom manipulator kinematics based on matlab toolbox," in *2017 International Conference on Advanced Mechatronic Systems (ICAMechS)*, 2017, pp. 376–380.

[12] MATLAB, *9.7.0.1190202 (R2019b)*. Natick, Massachusetts: The MathWorks Inc., 2018.

[13] "Pwm control of a dc motor used to drive a conveyor belt," *Procedia Engineering*, vol. 100, pp. 299 – 304, 2015, 25th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2014.

[14] A. P. Singh, U. Narayan, and A. Verma, "Speed control of dc motor using pid controller based on matlab," *Innovative Systems Design and Engineering*, vol. 4, pp. 22–28, 2013.

Appendix A

HARDWARE SPECIFICATION FOR WEB CAM AND 3D SCANNER

Kinova 4DoF Robot Manipulator

Weight : 5.2 kg

Payload : 0.8 kg (full extension) 1.3 kg (mid-range)

Arm reach : app. 70 cm

Maximum linear speed : 20 cm/s

Communication rate : 500 Hz

Camera Module

Resolution: 12.0 mega pixels (Software Enhanced)

Image Sensor: 1.3 Mega Pixels CMOS VGA Sensor

Max Dynamic Resolution: 1600 X 1200 pixels

Max Static Resolution: 3648 X 2736 pixels

Interface: USB 1.1/2.0 compatible

Frame Rate: 30 frames/second

Signal Noise Ratio: Larger than 48db

Dynamic Range: Larger than 72db

Brightness/Color: Auto Adjusting or Manual Mode (optional)

System Requirement: Windows ME / 2000 / XP / Vista(32-bit)

3D Scanner

Maximum power consumption: 5.0 VDC

Scan volume min: 0.2 m x 0.2 m x 0.2 m

Scan volume max: 2 m x 2 m x 2 m

Dimensions: 17.8 cm x 12.9 cm x 3.3 cm

Operating range: Min 0.4 m Max: 1.6 m

Field of view: Horizontal: 45°, Vertical: 57.5°, Diagonal: 69° Depth image size: 640 px (w) x 480 px (h)

Spatial x/y resolution @ 0.5 m: 0.9 mm

Depth resolution @ 0.5 m: 1 mm

Operating temperature: 10 - 40 °C

Data interface: USB 3.0

Appendix B

5TH ORDER POLYNOMIAL POSITION, VELOCITY AND ACCELERATION PLOTTING CODE

```
p0 = 0;
p1 = 4;

[p,pd,pdd] = tpoly(p0, p1, 60);
subplot(3,1,1); plot(p,'r'); xlabel('Time'); ylabel('Pos');
subplot(3,1,2); plot(pd,'g'); xlabel('Time'); ylabel('Vel');
subplot(3,1,3); plot(pdd,'b'); xlabel('Time'); ylabel('Acce');
hold on;

T0 = transl(0.4, 0.2, 0) * trotx(pi);
T1 = transl(-0.4, -0.2, 0.3) * troty(pi/2) * trotz(-pi/2);
T = ctraj(T0, T1, 50);
```

Appendix C

QUINTIC POLYNOMIAL TRAJECTORY GENERATION MATLAB CODE

```
% This code generates a minimum Jerk Trajectory for the Jaco arm using
%** Quintic Polynomial Trajectories **

% this code generates quintic polynomials between the way points - constant
% speed and zero
clear X XX T TT t x;
% intial time, joint value, joint velocity and joint acceleration
t_0=2.001;
t_end=60;
% making a circle in 60 sec?
X=zeros(10,4);%3 possition (x,y,z) and roll angle (alpha)
V = zeros(10,4);%3 linear speeds (Vx,Vy,Vz) and rotation about x
%define the way points and speeds between segments in each direction
period = t_end - t_0;%cycle time = given distance to complete the scan/conveyor speed
%-----------generate way points and EE speeds for a given/specified path-- ---------
p = 0:2*pi/20:2*pi;
x_c = 0.25*sin(p);
vx_c= 0.25*2*pi*cos(p)/period;

y_c = 0.25* cos(p);
vy_c = -0.25*2*pi*sin(p)/period;

for i = 1:length(X);
    X(i,:) = [x_c(i) y_c(i) 0.6 0];%positions [x_c(i) y_c(i) z_c(i) roll(i)]
    V(i,:)  = [vx_c(i) vy_c(i) 0 0.5];%speeds
end
%-------------------------------------------------------------------------------------
s=size(X);

%T = t_0:(t_end/(s(1)-1)-t_0)/500:t_end/(s(1)-1);

T = t_0:0.002:t_end/(s(1)-1);
```

```
% generating the whole time
for p=1:s(1)-1
    TT((p-1)*length(T)+1:p*length(T))=T+(p-1)*T(end);
end
for i=1:s(1) % for each waypoint
    t(i)=t_0+(i-1)*(t_end-t_0)/(s(1)-1);
    x(i)=X(i,1);
    y(i)=X(i,2);
    z(i)=X(i,3);
    r(i)=X(i,4);%roll angle
end
 for j=1:(s(1)-1)

     A = [1,  t(j),  t(j)^2, t(j)^3,   t(j)^4,    t(j)^5;   ...
     0,  1,   2*t(j), 3*t(j)^2, 4*t(j)^3,  5*t(j)^4;   ...
     0,  0,   2,    6*t(j),   12*t(j)^2, 20*t(j)^3; ...
     1,  t(j+1),  t(j+1)^2, t(j+1)^3,   t(j+1)^4,    t(j+1)^5;   ...
     0,  1,   2*t(j+1), 3*t(j+1)^2, 4*t(j+1)^3,  5*t(j+1)^4;   ...
     0,  0,   2,    6*t(j+1),   12*t(j+1)^2, 20*t(j+1)^3];
    % Vector of intial and final joint positions and velocities

   for k=1:4 % x y z r
    if j~=s(1)-1 && j~=1 % first and last points have always zero speed.
      %b = [xo; xdoto; xddoto; xf; xdotf; xddotf];
         b = [X(j,k); V(j,k); 0; X(j+1,k); V(j+1,k); 0];
    elseif j==1

          b = [X(j,k); 0; 0; X(j+1,k); V(j+1,k); 0];
    elseif j==s(1)-1

        b = [X(j,k);  V(j,k) ; 0 ; X(j+1,k); 0; 0];
    end
    a = A\b;
    xx     = a(1) + a(2)*(T+(j-1)*T(end)) + a(3)*(T+(j-1)*T(end)).^2 + a(4)*(T+(j-1)*T(end)).^3 +
             a(5)*(T+(j-1)*T(end)).^4 + a(6)*(T+(j-1)*T(end)).^5;  % between the two waypoints

    xxdot  = a(2) + 2*a(3)*(T+(j-1)*T(end)) + 3*a(4)*(T+(j-1)*T(end)).^2 + 4*a(5)*(T+(j-1)*T(end)).^3 +
             5*a(6)*(T+(j-1)*T(end)).^4;

    xxddot = 2*a(3) + 6*a(4)*(T+(j-1)*T(end)) + 12*a(5)*(T+(j-1)*T(end)).^2 + 20*a(6)*(T+(j-1)*T(end)).^3;

    XX(k,(j-1)*length(T)+1:j*length(T))=xx(:);
    XX_dot(k,(j-1)*length(T)+1:j*length(T))=xxdot(:);
   end
end

figure;
plot(TT,XX(1,:),'r'); hold on
plot(TT,XX(2,:),'b'); hold on;
plot(TT,XX(3,:),'g'); hold on;
plot(TT,XX(4,:),'k'); hold on;

% waypoints
for k = 1:4
    for i = 1:s(1)
     plot(t(i),X(i,k),'k.');
    end
end

Traj.time=TT;
Traj.signals.values=XX';
Traj.signals.dimentions=6;

Traj_dot.time=TT;
Traj_dot.signals.values=XX_dot';
Traj_dot.signals.dimentions=6;
```

Appendix D

PID CONTROLLER FOR DC MOTOR

```
clc;
clear;
close all
% DC motor constants
J=0.02;
b=0.2;
kt=0.02;
ke=0.02;
R=2;
L=0.4;
num=[kt];
den=[J*L J*R+b*L b*R+ke*kt ];
% PID
kp=70;
ki=170;
kd=5;
% PID transfer function
numPID=[kd kp ki];
denPID=[1 0];
num_DC_PID=conv(num,numPID);
den_DC_PID=conv(den,denPID);

[NumPID_CLP,DenPID_CLP]=cloop(num_DC_PID,den_DC_PID);
disp('Closed loop TF of DC motor with PID controller')
tf(NumPID_CLP,DenPID_CLP)
figure
step(NumPID_CLP,DenPID_CLP), grid on
```

Appendix E

MATLAB CODE FOR OBJECT ORIENTATION ANGLE MEASUREMENT

```
rotate = -20; %degrees
scaleFactor = 0.3;
tic;
I1 = rgb2gray(imread('0.jpg'));
% I2 = imresize(imrotate(I1, rotate), scaleFactor); % same image scalled and rotated
I2 = rgb2gray(imread('45.jpg'));% match with template

pointsx1 = detectSURFFeatures(I1);
pointsx2 = detectSURFFeatures(I2);

[features1, valid_pointsx1] = extractFeatures(I1, pointsx1);
[features2, valid_pointsx2] = extractFeatures(I2, pointsx2);

[indexPairs, matcmetric] = matchFeatures(features1, features2);

matchedPoints1 = valid_pointsx1(indexPairs(:,1));
matchedPoints2 = valid_pointsx2(indexPairs(:,2));

showMatchedFeatures(I1,I2,matchedPoints1,matchedPoints2);

[tform, inlierDistorted, inlierOriginal] = estimateGeometricTransform(...
   matchedPoints1, matchedPoints2, 'similarity');

Tinv  = tform.invert.T;

ss = Tinv(2,1);
sc = Tinv(1,1);
scaleRecovered = sqrt(ss*ss + sc*sc);
thetaRecovered = atan2(ss,sc)*180/pi;
toc;
fprintf('scale = %.1f, rotated angle = %.1f\n', scaleRecovered, thetaRecovered);
% fprintf('Angle error = %.1f%%\n', (abs(rotate) - thetaRecovered)*100 / abs(rotate));
clear all;
```