

**Development of an Automated Industrial Painting System with Optimized Quality  
and Energy Consumption.**

by

Muhammad Idrees

A thesis submitted to the  
School of Graduate and Postdoctoral Studies in partial  
fulfillment of the requirements for the degree of

**Master of Applied Science in Electrical and Computer Engineering**

Faculty of Electrical and Computer Engineering

University of Ontario Institute of Technology (Ontario Tech University)

Oshawa, Ontario, Canada

October 2023

© [Muhammad Idrees, 2023](#)

## THESIS EXAMINATION INFORMATION

Submitted by: **Muhammad Idrees**

**Master of Applied Science in Electrical and Computer Engineering**

<p><b>Thesis title:</b> Development of an Automated Industrial Painting System with Optimized Quality and Energy Consumption.</p>
---

An oral defense of this thesis took place on Sep 20, 2023, in front of the following examining committee:

**Examining Committee:**

Chair of Examining Committee	Dr. Mennatullah Siam
Research Supervisor	Dr. Hossam Gaber
Examining Committee Member	Dr. Ruth Milman
Thesis Examiner	Dr. Meaghan Charest-Finn

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

## **ABSTRACT**

Paint application is vital for product durability and aesthetics, whether done manually or by precise robotic systems. Manual work is error-prone and risky, while robots offer accuracy. However, programming robot trajectories for diverse products is challenging. Therefore, developing an autonomous system capable of generating automated paint trajectories is desirable. While adequate work has been done to optimize paint trajectories for coating thickness on complex free-form surfaces, the investigation of robot energy consumption and process time in the context of painting is left unattended. Thus, this study focuses on formulation of a hybrid optimization scheme to generate time and energy-efficient paint trajectories while ensuring optimal coating deposition on a surface. Moreover, considerable effort is put into the development of hardware and software for the integrated robotic system. Results for the trajectory optimization of a car door, hood, and bumper reveal efficient paint trajectories can be obtained using the proposed optimization scheme.

**Keywords:** trajectory optimization; 3D scanning; automation; Genetic Algorithm; ROS;

## **AUTHOR'S DECLARATION**

I hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize the University of Ontario Institute of Technology (Ontario Tech University) to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize University of Ontario Institute of Technology (Ontario Tech University) to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

A handwritten signature in black ink, reading "Muhammad Idrees", written in a cursive style and underlined.

---

Muhammad Idrees



## **STATEMENT OF CONTRIBUTIONS**

### **Part of this work described in Chapter 3 has been presented in:**

M. Idrees and H. A. Gabbar, "Automated 3D scanning system for extracting surface geometries," in *Symposium on Plasma and Nuclear Systems*, Oshawa, Canada, Aug 2023

### **Part of this work described in Chapter 3 has been presented in:**

M. Idrees and H. A. Gabbar, "Automated Surface Scanning for Industrial Applications," in *WORKSHOP ON SMART SCAN*, Oshawa, Canada, 2023.

### **Part of this work described in Chapter 4 is under review for:**

M. Idrees and H. A. Gabbar, "A hybrid optimization scheme for efficient trajectory planning of a spray-painting robot" in 3rd International Conference on Robotics, Automation, and Artificial Intelligence (RAAI), Singapore, Dec 14-16, 2023.

## **ACKNOWLEDGEMENTS**

Special thanks to Manir Isham and other lab members in the Smart Energy Systems Lab at Ontario Tech for helping in the structural fabrication of the integrated system. Thanks to the program supervisor, Dr. Hossam Gaber for supporting the thesis and project. Gratitude to the supervisory and examination committee for providing feedback on the manuscript. Finally, I would like to extend my gratefulness to Mitacs Accelerate and Cherkam Industrial Systems (Industrial partner) for sponsoring and funding the project.

## TABLE OF CONTENTS

<b>THESIS EXAMINATION INFORMATION</b> .....	<b>ii</b>
<b>ABSTRACT</b> .....	<b>1</b>
<b>AUTHOR'S DECLARATION</b> .....	<b>2</b>
<b>STATEMENT OF CONTRIBUTIONS</b> .....	<b>3</b>
<b>ACKNOWLEDGEMENTS</b> .....	<b>4</b>
<b>LIST OF TABLES</b> .....	<b>8</b>
<b>LIST OF FIGURES</b> .....	<b>10</b>
<b>LIST OF ABBREVIATIONS AND SYMBOLS</b> .....	<b>18</b>
<b>NOMENCLATURE</b> .....	<b>20</b>
<b>Chapter 1. Introduction</b> .....	<b>26</b>
1.1 Background and motivation .....	26
1.2 Problem definition and scope of the thesis .....	27
1.3 Outline of thesis .....	28
<b>Chapter 2. Literature Review</b> .....	<b>29</b>
2.1 Integrated systems for automated industrial painting .....	29
2.2 3D scanning and measurement .....	31
2.2.1 Passive 3D scanning methods .....	32
2.2.2 Active 3D scanning methods .....	33
2.2.3 Integrated systems for 3D scanning .....	35
2.3 Trajectory planning and optimization .....	36
2.3.1 Trajectory planning with paint quality optimization.....	36
2.3.2 Trajectory planning with energy optimization .....	39
2.4 Trajectory execution in simulation environment .....	40
2.4.1 CAD models.....	41
2.4.2 Spray-painting methods .....	41
2.4.3 Robot simulation software .....	42
2.5 Trajectory execution on industrial painting robots .....	43
2.5.1 Industrial painting robots .....	44
2.5.2 Robot programs.....	46
2.6 Validation of paint quality .....	47
2.7 Research objectives.....	48
<b>Chapter 3. Design of 3D Scanning System</b> .....	<b>49</b>
3.1 Introduction.....	49
3.2 Methodology .....	50

3.2.1	Mechanism for 3D scan acquisition.....	51
3.2.2	Depth map to point cloud.....	52
3.2.3	Box Filter to extract region of interest.....	53
3.2.4	Noise removal and raw alignment of point clouds.....	54
3.2.5	Fine alignment using ICP.....	55
3.2.6	CAD calibration in camera frame.....	57
3.2.7	Evaluating accuracy of the 3D scan.....	58
3.2.7.1	Pseudo code for computing $D_1$ metrics.....	59
3.2.7.2	Pseudo code for computing $D_2$ metrics.....	59
3.2.7.3	Pseudo code for computing $D_3$ metrics.....	60
3.2.7.4	Pseudo code for computing $A_3$ metrics.....	61
3.2.7.5	Pseudo code for evaluating 3D scan accuracy.....	61
3.3	Conclusion.....	62
<b>Chapter 4. Optimal Paint Trajectory Planning.....</b>		<b>63</b>
4.1	Introduction.....	63
4.2	Methodology.....	63
4.2.1	Establishment of spraying process model.....	63
4.2.2	Establishment of coating deposition model.....	65
4.2.3	Manipulator forward kinematics model.....	68
4.2.4	Manipulator inverse kinematics model.....	70
4.2.5	Manipulator velocity analysis and Jacobian.....	71
4.2.6	Manipulator acceleration analysis and Hessian.....	71
4.2.7	Manipulator torque and energy model.....	72
4.2.8	Hybrid optimization scheme.....	73
4.3	Conclusion.....	81
<b>Chapter 5. Integrated System Development.....</b>		<b>82</b>
5.1	Introduction.....	82
5.2	Methodology.....	82
5.2.1	Hardware development.....	84
5.2.2	Software development.....	88
5.2.3	Graphical user interface.....	90
5.2.4	ROS RQT graph.....	93
5.3	Conclusion.....	93
<b>Chapter 6. Results and Discussions.....</b>		<b>94</b>
6.1	Spraying process, robot, and optimizer parameters.....	94

6.2	3D scanning and CAD calibration results.....	97
6.3	Optimal paint trajectory planning for a car door.....	101
6.3.1	Results for slicing direction $\theta= 0^\circ$ .....	101
6.3.2	Results for slicing direction $\theta= 30^\circ$ .....	103
6.3.3	Results for slicing direction $\theta= 60^\circ$ .....	104
6.3.4	Results for slicing direction $\theta= 90^\circ$ .....	106
6.3.5	Results discussions.....	107
6.4	Optimal paint trajectory planning for a car hood.....	110
6.4.1	Results for slicing direction $\theta= 0^\circ$ .....	110
6.4.2	Results for slicing direction $\theta= 30^\circ$ .....	111
6.4.3	Results for slicing direction $\theta= 60^\circ$ .....	113
6.4.4	Results for slicing direction $\theta= 90^\circ$ .....	114
6.4.5	Results discussions.....	116
6.5	Optimal paint trajectory planning for a car bumper.....	118
6.5.1	Results for slicing direction $\theta= 0^\circ$ .....	118
6.5.2	Results for slicing direction $\theta= 30^\circ$ .....	119
6.5.3	Results for slicing direction $\theta= 60^\circ$ .....	121
6.5.4	Results for slicing direction $\theta= 90^\circ$ .....	122
6.5.5	Results discussions.....	124
6.6	Experimental validation of energy consumption .....	126
6.7	Results comparison with literature.....	129
<b>Chapter 7. Conclusion and Future Works.....</b>		<b>131</b>
<b>References.....</b>		<b>133</b>
<b>Appendices.....</b>		<b>140</b>
A1.	Paint system CAD drawings .....	140
A2.	Robot specifications .....	140
A3.	Intel Real sense D435 specifications.....	141
A4.	VL53L0X TOF sensor specifications .....	141
A5.	Linear actuators specifications.....	142

## LIST OF TABLES

### CHAPTER 2

Table 2.1:	Trajectory optimization results for U and V direction trajectories.	37
Table 2.2:	Energy consumption results for COMAU Racer robot.	40
Table 2.3:	Summary of commercial robot simulation software.	43
Table 2.4:	Industrial robots for paint applications.	46

### CHAPTER 4

Table 4.1:	DH Table for 4 DOF PRRR manipulator.	69
------------	--------------------------------------	----

### CHAPTER 5

Table 5.1:	Hardware components breakdown with component IDs, descriptions, and the corresponding CAD models.	84
------------	---	----

### CHAPTER 6

Table 6.1:	List of spraying process, robot, and optimizer parameters used in analysis.	95
Table 6.2:	Scanned models and their corresponding CAD calibrated in frame $\{C\}$ .	98

Table 6.3:	Similarity scores between the 3D scanned models and the corresponding CAD.	99
Table 6.4:	Results summary for trajectory planning and optimization of a car door for both equidistant and non-equidistant slicing.	109
Table 6.5:	Results summary for trajectory planning and optimization of a car hood for both equidistant and non-equidistant slicing.	117
Table 6.6:	Results summary for trajectory planning and optimization of a car bumper for both equidistant and non-equidistant slicing.	125
Table 6.7:	Results summary of experimental validation of energy consumption for optimal paint trajectories of car door, car hood and car bumper.	128
Table 6.8:	Results comparison summary with literature.	129

## LIST OF FIGURES

### CHAPTER 2

Figure 2.1:	Integrated system for robotic painting.	30
Figure 2.2:	Integrated system for paint quality validation.	31
Figure 2.3:	Classification of 3D scanning and measurement methods.	32
Figure 2.4:	Euclidean error for a statue and office scene. The point cloud is taken via a depth camera and registered in Kinect Fusion.	34
Figure 2.5:	Accuracy comparison of naive, L1diag, CSR and WT+CT.	34
Figure 2.6:	Integrated system for 3D reconstruction of objects.	35
Figure 2.7:	Coating thickness model for a complex free-form surface.	38
Figure 2.8:	Coating thickness results for transitional segment opt.	39
Figure 2.9:	KR AGILUS KR 10 R1100.	44
Figure 2.10:	P-250iB/15 by FANUC.	45
Figure 2.11:	IRB 5500 Flex Painter by ABB.	45
Figure 2.12:	Image processing pipeline for paint validation.	47



### CHAPTER 3

Figure 3.1:	(left) Intel Real Sense D435 sensor (right) VLX53LoX.	49
Figure 3.2:	Methodology for 3D scan acquisition.	50
Figure 3.3:	Schematic of 3D scan acquisition mechanism.	51
Figure 3.4:	Schematic of 2D pixel image and corresponding depth values.	52
Figure 3.5:	Box Filter on raw point cloud.	53
Figure 3.6:	Schematic of ICP alignment between source and target point cloud.	55
Figure 3.7:	Pipeline for evaluating accuracy of the 3D scan.	58

### CHAPTER 4

Figure 4.1:	Spraying torch model (Elliptical Paint area).	64
Figure 4.2:	Elliptical double beta distribution model of coating thickness on an elliptical surface area.	64
Figure 4.3:	Coating deposition model on a complex free-form surface.	66
Figure 4.4:	DH Schematic of a 4 DOF PRRR manipulator.	68
Figure 4.5:	Slicing model showing the sliced region, the elliptical paint area, and the trajectory points.	74

Figure 4.6: Trajectory planning and coating deposition model on a complex free-form surface with slice sandwiched between two spraying gun positions. 74

Figure 4.7: Trajectory planning and optimization algorithm. The input to the optimization algorithm is a CAD model while the output is an optimized trajectory for the paint robot in task space. The end-effector trajectory includes the x, y, z location, orientation, and the velocity vector at a given point in task space. 80

## CHAPTER 5

Figure 5.1: Software and hardware development methodology. 83

Figure 5.2: CAD schematic of the Integrated System with component IDs. 87

Figure 5.3: 3D rendered CAD model of the Integrated System (isometric view). 87

Figure 5.4: 3D rendered CAD model of the Integrated System (top and side view). 88

Figure 5.5: Schematic for Software development of the integrated system for automated industrial painting with optimized paint quality and energy consumption. 89

Figure 5.6: Front panel of web-based GUI. 90

Figure 5.7: Web GUI Optimizer Settings and File System Handler. 91

Figure 5.8: Web GUI miscellaneous buttons and functions. 92

Figure 5.9:	Software packages and custom Python scripts.	92
Figure 5.10:	ROS RQT Graph.	93
 <b>CHAPTER 6</b>		
Figure 6.1:	Experimental setup in laboratory.	95
Figure 6.2:	Selecting 3D scan and viewing it in the GUI.	97
Figure 6.3:	Calibrating the 3D scan and the corresponding CAD file in the GUI.	98
Figure 6.4:	$D_1$ and $D_2$ density distributions for car door.	100
Figure 6.5:	$D_1$ and $D_2$ density distributions for car hood.	100
Figure 6.6:	$D_1$ and $D_2$ density distributions for car bumper.	100
Figure 6.7:	Coating thickness and planned trajectory of a car door in $\{EF\}$ for slicing direction $\theta = 0^\circ$ (left: equidistant slicing, right: non-equidistant slicing).	101
Figure 6.8:	Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car door: $\theta = 0^\circ$ ).	102
Figure 6.9:	Coating thickness and planned trajectory of a car door in $\{EF\}$ for slicing direction $\theta = 30^\circ$ (left: equidistant slicing, right: non-equidistant slicing).	103
Figure 6.10:	Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car door: $\theta = 30^\circ$ ).	104

Figure 6.11:	Coating thickness and planned trajectory of a car door in {EF} for slicing direction $\theta = 60^\circ$ (left: equidistant slicing, right: non-equidistant slicing).	104
Figure 6.12:	Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car door: $\theta = 60^\circ$ ).	105
Figure 6.13:	Coating thickness and planned trajectory of a car door in {EF} for slicing direction $\theta = 90^\circ$ (left: equidistant slicing, right: non-equidistant slicing).	106
Figure 6.14:	Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car door: $\theta = 90^\circ$ ).	107
Figure 6.15:	Total energy, trajectory time, coating deviation and relative coating error vs slicing direction (car door).	108
Figure 6.16:	Coating thickness and planned trajectory of a car hood in {EF} for slicing direction $\theta = 0^\circ$ (left: equidistant slicing, right: non-equidistant).	110
Figure 6.17:	Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car hood: $\theta = 0^\circ$ ).	111
Figure 6.18:	Coating thickness and planned trajectory of a car hood in {EF} for slicing direction $\theta = 30^\circ$ (left: equidistant slicing, right: non-equidistant slicing).	111

Figure 6.19:	Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car hood: $\theta = 30^\circ$ ).	112
Figure 6.20:	Coating thickness and planned trajectory of a car hood in {EF} for slicing direction $\theta = 60^\circ$ (left: equidistant slicing, right: non-equidistant slicing).	113
Figure 6.21:	Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car hood: $\theta = 60^\circ$ ).	114
Figure 6.22:	Coating thickness and planned trajectory of a car hood in {EF} for slicing direction $\theta = 90^\circ$ (left: equidistant slicing, right: non-equidistant slicing).	114
Figure 6.23:	Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car hood: $\theta = 90^\circ$ ).	115
Figure 6.24:	Total energy, trajectory time, coating deviation and relative coating error vs slicing direction (car hood).	116
Figure 6.25:	Coating thickness and planned trajectory of a car bumper in {EF} for slicing direction $\theta = 0^\circ$ (left: equidistant slicing, right: non-equidistant slicing).	118
Figure 6.26:	Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car bumper: $\theta = 0^\circ$ ).	119

Figure 6.27:	Coating thickness and planned trajectory of a car bumper in {EF} for slicing direction $\theta = 30^\circ$ (left: equidistant slicing, right: non-equidistant slicing).	119
Figure 6.28:	Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car bumper: $\theta = 30^\circ$ ).	120
Figure 6.29:	Coating thickness and planned trajectory of a car bumper in {EF} for slicing direction $\theta = 60^\circ$ (left: equidistant slicing, right: non-equidistant slicing).	121
Figure 6.30:	Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car bumper: $\theta = 60^\circ$ ).	122
Figure 6.31:	Coating thickness and planned trajectory of a car bumper in {EF} for slicing direction $\theta = 90^\circ$ (left: equidistant slicing, right: non-equidistant slicing).	122
Figure 6.32:	Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car bumper: $\theta = 90^\circ$ ).	123
Figure 6.33:	Total energy, trajectory time, coating deviation, and relative coating error vs slicing direction (car bumper).	124
Figure 6.34:	Experimental energy consumption for trajectory optimization of car door.	126
Figure 6.35:	Experimental energy consumption for trajectory optimization of a car hood.	127

Figure 6.36: Experimental energy consumption for trajectory optimization of car bumper. 127

Figure 6.37: Robot executing trajectory on a car door. 128

## LIST OF ABBREVIATIONS AND SYMBOLS

LIDAR	Light Detection and Ranging
LASER	Light Amplification by Stimulated Emission of Rays
RGB	Red, Green, Blue
RGBD	RGB and depth image
PCA	Principal Component Analysis
ICP	Iterative Closest Point
SVD	Singular Value Decomposition
CMM	Coordinate Measuring Machine
CT	Computed Tomography
RFDIC	Rotation-Free Digital Image Correlation
FSM	Fan Shape Model
GA	Genetic Algorithm
RMS	Root Mean Square
CAD	Computed Aided Design
CAS	Computer Assisted Software
STL	Stereo Lithography
HVLP	High Volume Low Pressure
IRPS	Integrated Robotic Painting System
GUI	Graphical User Interface
ROS	Robot Operating System
TOF	Time of Flight
SQP	Sequential Quadratic Programming
<i>PFeatureDetector</i>	Process Oriented Feature Detector
IWO	Invasive Weed Optimization
TP	Teach Pendant
LS	List format file
JS	JavaScript
HTML	Hyper-text Markup Language
CSS	Cascading Style Sheets



GPIO	General Purpose Input Output
LINUX	Lovable Intellect Not Using XP
RNE	Recursive Newton Euler
HSI	Hue, Saturation, Intensity
DH	Denavit-Hartenberg
VOC	Volatile Organic Compounds
FEA	Finite Element Analysis
RVIZ	ROS visualization
TF	Transform
ROI	Region of Interest
URDF	Unified Robot Description Format
OS	Operating System

## NOMENCLATURE

### CHAPTER 2

$D_1$	Distance between randomly sampled point and the centroid of a point cloud
$D_2$	Distance between two randomly sampled points in a point cloud
$D_3$	Area of triangle formed by 3 randomly sampled points in a point cloud
$A_3$	The angle of the vertex of triangle formed by 3 randomly sampled points in a point cloud
$\Delta M_{color}$	Amount of color deposited on an incremental area $\Delta S$
$\Delta S$	Incremental area
$\varphi(\alpha)$	Paint flow function per unit area $[\frac{kg}{m^2s}]$
$\delta_{color}$	Surface color density $[\frac{kg}{m^2}]$
$i_{color}$	Color intensity $[HSI]$
$k_c$	Color constant
$i_{scan}$	Intensity range between 0 and 1 $[HSI]$
$k_s$	Scan constant
$\phi(\alpha)$	Color flow function $[\frac{HSI}{s}]$
$\phi(r)$	Gaussian approximation of color flow function

### CHAPTER 3

$\{C\}$	Depth Camera frame
$\{S\}$	Depth Sensor frame
$\{S_1\}$	Depth sensor frame translated to axis of rotation
$\{S_{1r}\}$	$\{S_1\}$ frame rotated with the axis of rotation
$\{cg\}$	Centroid frame of the point cloud with its axes aligned with the depth camera frame $\{C\}$

$P^C$	Point cloud in frame the depth camera frame $\{C\}$
${}^C_S T$	A 4 by 4 homogenous transform between frame $\{C\}$ and $\{S\}$
${}^{S_1} T$	A 4 by 4 homogenous transform between frame $\{S\}$ and $\{S_1\}$
$f_x, f_y$	Focal length of depth camera in x and y direction
$c_x, c_y$	Depth Camera center in the world frame
$u, v$	Pixel coordinates
$S$	Skew between world frame $\{W\}$ and camera frame $\{C\}$
$l_x, l_y, l_z$	Box filter length, width, and height
$\{S_2\}$	Frame at the centroid of the box filter with its axes aligned with the camera frame $\{C\}$
$P^{S_1(i)}$	Represents the $i_{th}$ point cloud in frame $\{S_1\}$
$P_{aligned}^{C(i)}$	Represents the $i_{th}$ point cloud aligned in the camera frame $\{C\}$
$P_{scan}^C$	Scanned model point cloud in the camera frame $\{C\}$
$P_{scan}^{eig}$	Scanned point cloud in the eigen coordinate frame
$P_{CAD}$	CAD point cloud in some arbitrary reference frame
$P_{CAD}^{eig}$	CAD point cloud in the eigen coordinate frame
$u_{CAD}$	A 3 by 3 eigen matrix for the CAD point cloud
$u_{scan}$	A 3 by 3 eigen matrix for the scanned point cloud
$c_{CAD}$	Principal center of the CAD point cloud
$c_{scan}$	Principal center of the scanned point cloud
$P_{CAD}^{aligned}$	CAD point cloud aligned in the eigen frame of the scanned point cloud
$P_{CAD}^C$	CAD point cloud aligned in the camera frame $\{C\}$
$R_y(\theta_{y(i)})$	Represents the 4 by 4 homogenous transformation matrix giving a relative rotation of $\theta_y$ degree about the y axis
$N_{pcd}$	Number of points in a point cloud
$q_i$	Individual point in a target point cloud for ICP alignment

$p_i$	Individual point in a source point cloud for ICP alignment
$n_{q_i}$	Normal vector at a given point $q_i$ in the target point cloud
$R, t$	A 3 by 3 rotation matrix and a 3 by 1 translational vector obtained by ICP
$p2p$	Short notation for point-to-point ICP
$p2plane$	Short notation for point-to-plane ICP
$P_{merged}^C$	Merged point cloud in the camera frame {C}

## CHAPTER 4

$a$	Longer side of ellipse
$b$	Shorter side of ellipse
$h$	Perpendicular height of the spray gun from the surface of the object
$\beta_x$	Beta value representing the spread of coating thickness along the X direction of ellipse
$\beta_y$	Beta value representing the spread of coating thickness along the Y direction of ellipse
$\varphi_x^{(max)}$	The maximum opening angle (torch angle) of ellipse in X direction
$\varphi_y^{(max)}$	The maximum opening angle (torch angle) of ellipse in Y direction
$Q_{C_1}$	Paint flow rate at point $C_1$
$Q_{C_2}$	Paint flow rate at point $C_2$
$A_{C_1}$	Paint area at point $C_1$
$A_{C_2}$	Paint area at point $C_2$
$d_{C_1}$	Coating thickness at point $C_1$
$d_{C_2}$	Coating thickness at point $C_2$
$L_s$	Connection line between the paint gun and a point $s$ on the surface of the object
$\bar{n}$	Normal vector of the surface at point $s$
$M_1$	Tangent plane at point $O$ on the surface

$M_2$	A parallel plane to $M_1$ intercepting point $s$ on the surface
$\gamma$	Angle between normal vector $\bar{\mathbf{n}}$ and $L_s$
$L_1, L_2, L_3$	Link lengths of the 4 DOF PRRR manipulator
$m_1, m_2, m_3$	Link masses of the 4 DOF PRRR manipulator
$M$	Link 0 mass (robot base mass)
$M(q)$	Manipulator inertia matrix
$C(q, \dot{q})$	Manipulator Coriolis and centrifugal acceleration matrix
$H(q, \dot{q})$	Manipulator gravity and friction dynamics matrix
$d$	Vertical offset of the robot base
$q$	Joint space angles vector
$\dot{q}$	Joint space velocity vector
$\dot{x}$	Task space velocity vector
$\ddot{q}$	Joint space acceleration vector
$\ddot{x}$	Task space acceleration vector
$J$	Jacobian in frame $\{0\}$
$H$	Hessian in frame $\{0\}$
$\tau_n$	Torque at joint $n$
$\omega_n$	Angular velocity of link $n$
$N_{joints}$	Number of joints in a serial-link manipulator
$P_{mech}$	Total mechanical power of a manipulator
$E_{AB}$	Energy consumed by the manipulator while moving from point A to B
$t_{AB}$	Time taken by the manipulator while moving from point A to B
$\{SF\}$	Slicing coordinate frame
$\{EF\}$	Eigen coordinate frame
$v_{(i)}$	Speed of the paint gun along a given slicing plane $i$
$x_{(i)}$	The x coordinate of the ellipse used in the coating function at a given slicing plane $i$
$\cos \gamma_{(i)}$	Cosine of angle $\gamma$ at a given slicing plane $i$
$\cos \varphi_{x(i)}$	Cosine of angle $\varphi_x$ at a given slicing plane $i$
$h_{s(i)}$	Parameter $h_s$ at a given slicing plane $i$

$p^{(i,j)}$	Trajectory coordinates at a slicing plane $i$ and trajectory point $j$ represented in the slicing frame {SF}
$\psi^{(i,j)}$	End-effector orientation vector at a slicing plane $i$ and trajectory point $j$ represented in the slicing frame {SF}
$\bar{v}^{(i,j)}$	Velocity vector at a slicing plane $i$ and trajectory point $j$ represented in the slicing frame {SF}
$P_{patch}$	Point cloud of a patch $P_{patch} \in \mathbb{R}^{(3, N_{patch})}$
$N_{patch}$	Number of points in a patch
$N_{pts}$	Number of points in a slice
$p^{robot}$	Trajectory coordinates in the robot frame {0}: $p^{robot} \in \mathbb{R}^{(3, N_t)}$
$\psi^{robot}$	Orientation vectors in the robot frame {0}: $\psi^{robot} \in \mathbb{R}^{(3, N_t)}$
$V^{robot}$	Velocity vectors in the robot frame {0}: $V^{robot} \in \mathbb{R}^{(3, N_t)}$
$d_{ideal}$	Desired coating thickness
$d_s$	Coating thickness at a point $s$ on the surface
$d_{mean}$	Mean coating thickness over a region of surface
$d_{std}$	Standard deviation of coating thickness over a region of surface
$N_t$	Total trajectory points in a slice
$\Delta T^{(n_t)}$	Time delta between two trajectory points
$J_{d_s}$	Mean squared error coating cost function
$J_{d_{error}}$	Coating deviation cost function
$J_E$	Mean energy cost function.
$J_T$	Mean trajectory time cost function
$J_{tot}$	Total cost function
$\omega_1$	Scaling factor for mean squared error cost
$\omega_2$	Scaling factor for coating deviation cost
$\omega_3$	Scaling factor for energy cost
$\omega_4$	Scaling factor for time cost
$\epsilon$	Hyper parameter in the cost function
$\delta$	Slice width
$v_{min}$	Minimum speed of the spraying gun

$v_{max}$	Maximum speed of the spraying gun
$ik_{cf}$	An integer index representing the inverse kinematic configuration when converting task space coordinates to joint space
$\theta$	Slicing direction: Rotation angle between frame {EF} and {SF}

## CHAPTER 6

$I_{avg}$	Average current of the robot for the entire trajectory
$E_{sum}$	Total sum of energy across the trajectory points
$E_{sav}$	Percentage of energy savings
$d_{mean}$	Mean coating thickness on the entire surface of a point cloud
$d_{std}$	The standard deviation of the coating thickness on the entire surface of a point cloud
$d_{error}$	Ratio of $d_{std}$ and $d_{mean}$
$r_m$	Mutation rate in GA
$c_{type}$	Crossover type in GA
$m_{type}$	Mutation type in GA
$N_{parents}$	Number of mating parents in GA
$N_{gen}$	Number of generations in GA
$N_{sol}$	Number of solutions per population in GA

## **Chapter 1. Introduction**

### **1.1 Background and motivation**

Industrial painting has become increasingly important in modern manufacturing processes. The application of paint to a product's surface improves its longevity and aesthetics. When paint is applied to a surface, it not only increases the corrosion resistance of the surface, but also enhances its heat resistance, electrical insulation, and reactivity to harmful chemicals. Robotics play an important role in paint processes since they increase the process efficiency, productivity, and quality of the painted surface. Robotic systems can work continuously without the need for any breaks thereby, accelerating production times and reducing labor costs. According to a survey, vehicle production will increase to 111.7 million units by the year 2023 [1]. The increase in production rates of vehicles demands the automation of paint processes and a need to develop a fully autonomous system.

The process of paint automation is an ongoing topic in both academia and industry. The key technology in paint process automation is trajectory planning over the surface of a geometric model. Trajectory planning refers to finding an optimal paint gun path and velocity vectors while ensuring coating uniformity over the surface of an object. While much work has been done to develop trajectory optimization schemes to achieve coating uniformity over the finished surface, these methods do not consider the dynamics of the robot which leads to suboptimal trajectory planning. This thesis, therefore, focuses on formulating a hybrid trajectory optimization scheme utilizing a genetic algorithm by taking into consideration the geometry of the object, the dynamics of the spray-painting process, and the robot moving the paint gun.



## **1.2 Problem definition and scope of the thesis**

The manual painting process for industrial parts exhibits challenges related to variations in coating quality, extended production timelines, heightened environmental impact through VOC emissions [2], and compromised worker safety [3]. This thesis seeks to investigate and implement specific strategies such as automation, eco-friendly coating formulations, and process optimization to rectify these issues. Optimal paint trajectory planning requires an accurate model of the geometry of the object, the dynamics of the spraying process, and the robot moving the spraying gun. Thus, the scope of the thesis can be divided into four folds. First, to obtain the geometry of the object, a 3D scan acquisition system is developed to accurately measure the surface profile of the surface to be painted. Secondly, the spray paint profile and paint deposition model are established on a complex free-form surface and thirdly, an optimization algorithm for the optimal trajectory planning of the spray paint process subject to paint spray and robot dynamics is developed. Finally, the mathematical formulation of the proposed scheme is implemented in Python programming language and the energy consumption is validated experimentally. Additionally, a web-based GUI (graphical user interface) is also developed that lets the user interact with the integrated system to perform 3D scans on objects, optimally plan trajectory on the surface of the object, and execute the trajectory in real-time on the two robotic arms installed onboard.

### **1.3 Outline of thesis**

The thesis report is divided into 7 chapters. Chapter (1) describes the background of industrial spray paint processes and the motivation to continue this thesis study. It also describes the key concepts needed to achieve the goals of the study. Chapter (2) discusses the theoretical concepts needed to formulate research methodology by overcoming important shortcomings in literature. This includes the investigation of integrated systems used for industrial painting, 3D scanning techniques for acquiring the geometries of objects and generating signatures of 3D surfaces, trajectory planning, and optimization techniques for complex free-form surfaces, and finally, validation techniques for validating the uniformity of the deposited paint on the surface and energy consumed by the robot. Chapter (3) describes the development of the 3D scan acquisition system in detail. This includes the selection of hardware components and the application of software to generate a complete 3D scan of a complex free-form surface. Chapter (4) discusses the mathematical formulation of the paint spray profile, the paint deposition model on a complex free-form surface, and the optimization algorithm for obtaining an optimal trajectory for the paint process. Chapter (5) discusses the design and development of an integrated system for automating the painting process with details on the web-based GUI and the software components used. Chapter (6) discusses the results of the 3D scan acquisition system and 3D profile signatures for evaluating the accuracy of 3D scans, simulation results for the paint surface quality achieved and the energy consumed by the robot to verify the proposed optimization scheme, and finally, validation of the optimal trajectory executed online on the integrated system. Chapter (7) discusses the conclusions and the future recommendations of the thesis study by summarizing all the chapters.

## **Chapter 2. Literature Review**

The problems associated with manual painting could be addressed using an integrated robotic system capable of autonomously applying paint over surfaces. Such a system must contain all the necessary hardware and software components to achieve the desired automation. This includes hardware components such as an industrial robot, a paint delivery system, a 3D scanner, and a central processing unit. The software components include: a simulation or co-simulation environment for a friendly user interface, an algorithm for 3D scanning of the object, a blueprint for trajectory planning and optimization, and an execution mechanism for uploading the trajectory to the robot. The literature review will discuss in detail the current technologies used for industrial painting; the shortcomings associated with them, and finally suggest improvements to make the paint process more autonomous and efficient.

### **2.1 Integrated systems for automated industrial painting**

Literature research shows the crucial components in the integrated system design are a paint booth, a robotic system, and the required software collection. While designing an integrated system for paint automation, researchers focus on improving the coating uniformity, process times, and paint waste. An early 1980's integrated system for painting contains a paint booth, a robot apparatus, and a rail mechanism for moving the robots [4]. The main goal behind the development of the robotic system was to minimize paint waste by using precise robotic movements.

Similarly, a software and hardware-based prototype of an integrated robotic painting system is developed [5]. The software modules contain part designs, process planning, trajectory generation of robots, and motion control. The hardware components include a work cell controller, motor drives, robotic manipulator, surface scanner, and paint delivery units. The scanning interface uses a mechanical probe to get the topography of the surface to be painted and converts it into a CAD model. Conversely, a CAD model can directly be imported from the CAD library. It is then processed to generate the robot trajectories followed by their execution on the work cell controller.

Another integrated system developed uses an algorithm to model the spray-painting process, and a computer program to simulate a robot for painting curved surfaces [6]. The painting program makes it possible to find out the optimum parameters for spray painting such as the paint gun velocity, spray distance, and multiple paint paths. The modeling part is done using a CAS (computer-assisted software) by two methods. If the part is simple, a CAD model is generated in the software otherwise a laser scanner is used to get the 3D model. An algorithm is then formulated to perform the paint thickness analysis and the paint process is simulated. For the validation of the coating thickness, a flat surface is used. The paint is deposited in a single paint stroke and the coating thickness is measured using an ELCOMETER. The experimental setup includes a *FANUC ArcMater Sr. Industrial* robot, a BINKS 95-A spray gun [7], and an ELCOMETER 345 coating thickness gauge [8]. A schematic of this integrated system is shown in Fig. 2.1.

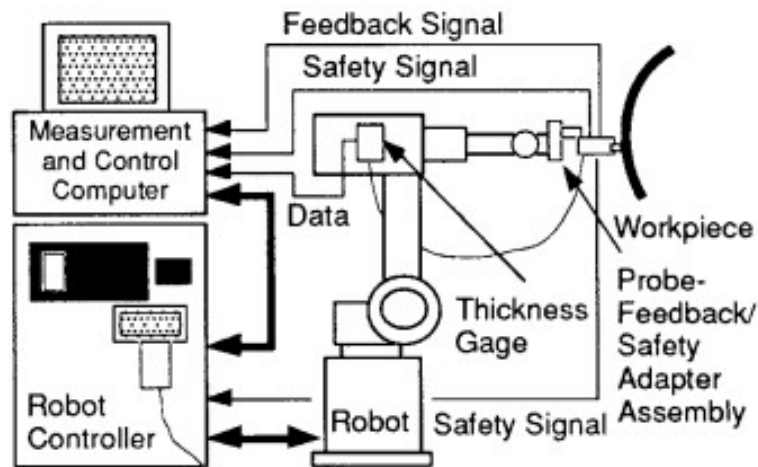


Figure 2.1: Integrated system for robotic painting. [6]

An integrated system containing a UR-10 robot with an HP-M2 airbrush [9] and a scanner for digitizing the paint intensity is developed [10]. It uses a blueprint to compute the coating intensities and then experimentally validates it. A single paint stroke is performed to deposit some amount of paint on a surface. After an image of the paint area is taken and digitized into color intensities, an image processing pipeline is applied to validate the coating thickness at each intensity point. The UR-10 robot with the spray gun is shown in Fig. 2.2.



Figure 2.2: Integrated system for paint quality validation. [10]

## 2.2 3D scanning and measurement

3D scanning is important for digitally recreating physical objects which can be accomplished using several methods. The most popular methods include CMM (Coordinate-measuring machine), laser scanners, and commercial computed tomography (CT) scanners [11]. CMM does one measurement at a time and therefore, is time consuming and less efficient. It also uses conventional monitoring equipment which makes it unsuitable for fast scanning. On the other hand, laser scanners are fast and can quickly scan the objects [12, 13, 14]. 3D scanning is also used for generating scenes for movies and games. In movies and games, 3D scanning is applied to objects, landscapes, and persons. It also finds its use in the screening of historical locations and objects for academic research. Upon a full 3D scan of a structure, it helps identify its integrity. 3D scanners can measure precise details in an object and captures complex geometries in a point cloud format [15, 16, 17, 18]. 3D scanning is utilized for reconstructing historical artifacts [19]. The early Aboriginal trackways discovered at the Wallenda Lakes World Heritage Site are used as a case study. 3D scanning can be broadly divided into two categories: passive and active 3D scanning as shown in Fig. 2.3.

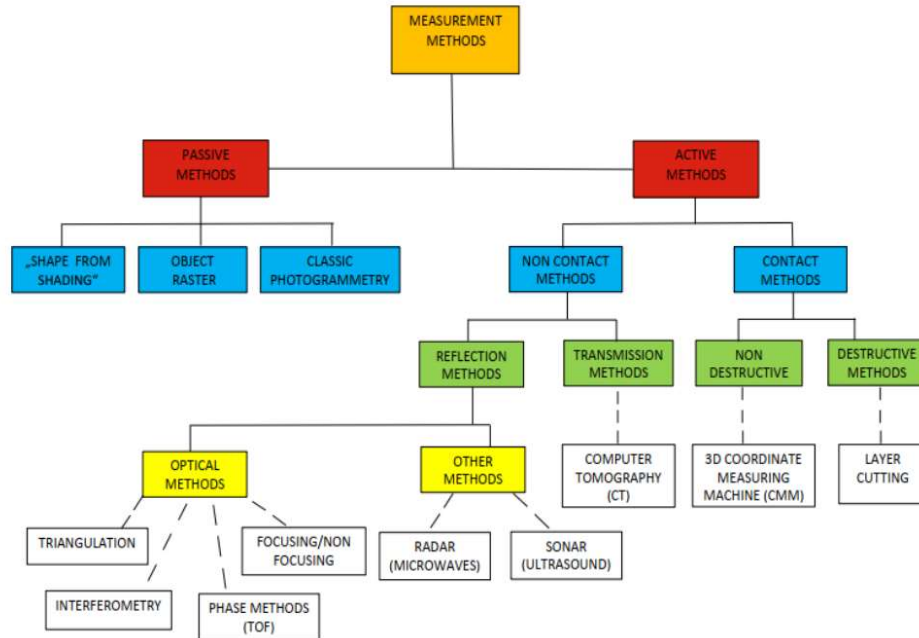


Figure 2.3: Classification of 3D scanning and measurement methods. [20]

### 2.2.1 Passive 3D scanning methods

Passive 3D scanning methods do not need a physical contact of the measuring device with the object. These methods use geometrically correct or stochastic markings on the surface combined with optical laws to capture the geometry of an object. Light based passive sensors work by using the reflected light from the surface of the object [20]. These methods are further divided into three categories: Shape from Shading, Object Raster, and Classic Photogrammetry. Shape from shading uses a surface image to compute the three-dimensional model of an object. A novel algorithm for shape from shading with multiple input images, realistic camera models, low angles of illumination, and uncertain camera positions is developed to capture a three-dimensional view of planetary images [21]. Raster scanning is the process of scanning line by line to cover an area. The most important of the passive methods is the photogrammetry. It uses overlapping photographs to create a 3D representation of an object. A researcher uses target-free photogrammetry to generate dense point clouds of different objects [22]. This algorithm uses a Rotation-Free Digital Image Correlation (RFDIC) method to improve the matching precision and a coarse-to-fine strategy to establish a multi-view geometry.

### 2.2.2 Active 3D scanning methods

Among the active 3D scanning methods, two subcategories are identified including contact-based and noncontact-based methods. The contact-based methods, require a physical contact between the measuring sensor and the object. A widely used approach is CMM which uses a mechanical probe to generate surface topography. CMM is an obsolete method and is slow compared to new optical based methods that use the principles of TOF (Time of Flight), Triangulation and Interferometry. LIDAR and LASER scanners are the prominent technologies when it comes to noncontact-based methods. A LIDAR based scanner is commonly used for mapping surroundings due to its long range. A LASER scanner on the other hand though limited in range, can capture more details of an object.

Kinect Fusion is a real-time mapping system used to capture indoor 3D scenes with variable lighting conditions by employing the use of a low-cost depth camera [23]. A per vertex Euclidean error and per vertex angle error metrics are used to analyze the accuracy of the Kinect Fusion method for 3D scenes [24]. It is observed that the Euclidean error lies within 0-15 mm for an office scene and within 0-8 mm for a statue as shown in Fig. 2.4. Similarly, a sparse reconstruction-based technique is used to generate a 3D environment using a few depth scans [25]. Since most of the surfaces and edges have regularity, this makes it possible to achieve high reconstruction accuracy using a limited number of measurements of the unknown environment. The results for the 3D reconstruction accuracy are shown in Fig. 2.5.

Another study aims at the comparison of different 3D scanning devices to capture a human face in 3D [26]. The accuracy is computed using mean squared error between the ground truth CAD model and the generated 3D scan. Such methods are commonly termed as surface registration-based techniques dependent on ICP error for accuracy evaluation [27]. ICP is an iterative process and uses the entire surface which makes it computationally inefficient. On the other hand, feature matching-based methods use mathematical transformations to obtain higher dimensional features of the surface for calculating accuracy [28]. One such method uses geometric signatures to encode the surface into  $D_1$ ,  $D_2$ ,  $D_3$ , and  $A_3$  features [29]. These features are probability distributions of measurements taken from the geometric model such as the distance between the centroid and points ( $D_1$ ),

the distance between two points ( $D_2$ ), the square root of the area of a triangle formed by 3 points ( $D_3$ ) and the angle formed by the vertex of 3 points ( $A_3$ ).

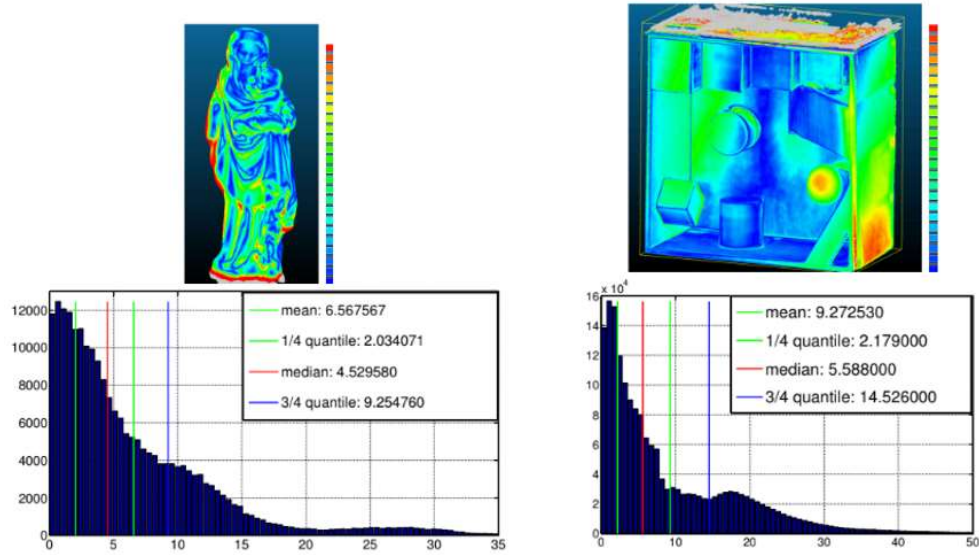


Figure 2.4: Euclidean error for a statue and office scene. The point cloud is taken via a depth camera and registered in Kinect Fusion. [24]

Name	Method	PSNR (dB) / Time (s) (Percentage of Samples)			
		0.5%	1%	5%	10%
Aloe	CSR	N/A	N/A	21.4 / 10.6	24.1 / 8.65
	WT+CT	N/A	N/A	21.9 / 19.4	24.3 / 19.5
	naive	N/A	21.6 / <b>0.17</b>	24.7 / <b>0.14</b>	26.0 / <b>0.22</b>
	Lldiag	<b>20.6 / 14.5</b>	<b>21.7 / 7.02</b>	<b>24.9 / 3.12</b>	<b>26.4 / 2.06</b>
Art	CSR	N/A	N/A	23.1 / 11.9	25.3 / 9.80
	WT+CT	N/A	N/A	25.0 / 19.8	26.7 / 19.5
	naive	21.9 / <b>0.15</b>	23.5 / <b>0.16</b>	26.3 / <b>0.17</b>	27.7 / <b>0.18</b>
	Lldiag	<b>22.5 / 11.1</b>	<b>23.8 / 8.86</b>	<b>26.6 / 3.78</b>	<b>27.8 / 2.23</b>
Baby	CSR	N/A	N/A	26.6 / 10.0	31.1 / 9.11
	WT+CT	N/A	24.1 / 19.6	27.7 / 19.4	31.5 / 19.5
	naive	27.6 / <b>0.15</b>	27.4 / <b>0.16</b>	31.3 / <b>0.16</b>	33.3 / <b>0.18</b>
	Lldiag	<b>27.8 / 12.1</b>	<b>28.4 / 10.5</b>	<b>32.5 / 3.21</b>	<b>33.9 / 2.06</b>
Dolls	CSR	N/A	N/A	24.3 / 13.2	26.5 / 11.0
	WT+CT	N/A	20.6 / 19.5	27.5 / 19.6	28.2 / 20.3
	naive	25.8 / <b>0.13</b>	24.5 / <b>0.16</b>	27.8 / <b>0.16</b>	28.5 / <b>0.18</b>
	Lldiag	<b>26.9 / 7.07</b>	<b>27.5 / 5.49</b>	<b>28.3 / 2.24</b>	<b>28.9 / 3.03</b>
Moebius	CSR	N/A	N/A	23.6 / 11.9	26.1 / 10.5
	WT+CT	N/A	22.4 / 19.3	26.3 / 19.5	27.6 / 19.4
	naive	25.7 / <b>0.14</b>	24.7 / <b>0.16</b>	26.8 / <b>0.15</b>	27.8 / <b>0.18</b>
	Lldiag	<b>25.8 / 6.91</b>	<b>26.4 / 7.03</b>	<b>27.5 / 2.90</b>	<b>28.6 / 2.59</b>
Rocks	CSR	N/A	N/A	23.1 / 11.5	25.0 / 9.15
	WT+CT	N/A	N/A	23.2 / 19.3	25.6 / 19.2
	naive	21.7 / <b>0.15</b>	23.8 / <b>0.15</b>	25.8 / <b>0.15</b>	27.2 / <b>0.19</b>
	Lldiag	<b>22.7 / 12.0</b>	<b>24.3 / 9.71</b>	<b>25.9 / 3.22</b>	<b>27.3 / 2.68</b>

Figure 2.5: Accuracy comparison of naive, L1diag, CSR, and WT+CT. [25]



### 2.2.3 Integrated systems for 3D scanning

An integrated system for 3D scanning comprises of hardware and software components to capture the 3D representation of a physical object. A 3D scanner is an important hardware component in realizing the 3D model of an object. It can be placed on a stationary mount or a movable mount like a robotic arm. A line profile laser scanner, an industrial robot, and a turntable mechanism are used to generate 3D scans of objects and convert them to CAD models [30]. A set of curves are defined around the volume along which the line profile scanner moves to generate a 3D scan of the object.

Similarly, 3D scanned models are investigated for contour tracing by designing a robotic system [31]. This system uses a 6-DOF robotic arm, a short-range laser scanner with 100 to 200 mm range, a 30  $\mu\text{m}$  resolution, and a turntable for rotating the work piece as shown in Fig. 2.6. The laser scanner can communicate with the computer and the robot controller. A similar scanning system for large-scale objects is proposed which uses a laser scanner, a turntable mechanism, and a robot for calibration of the system [32]. Nevertheless, a robotic system for surface measurement via a 3D scanner uses similar components to achieve the scanning task [33].

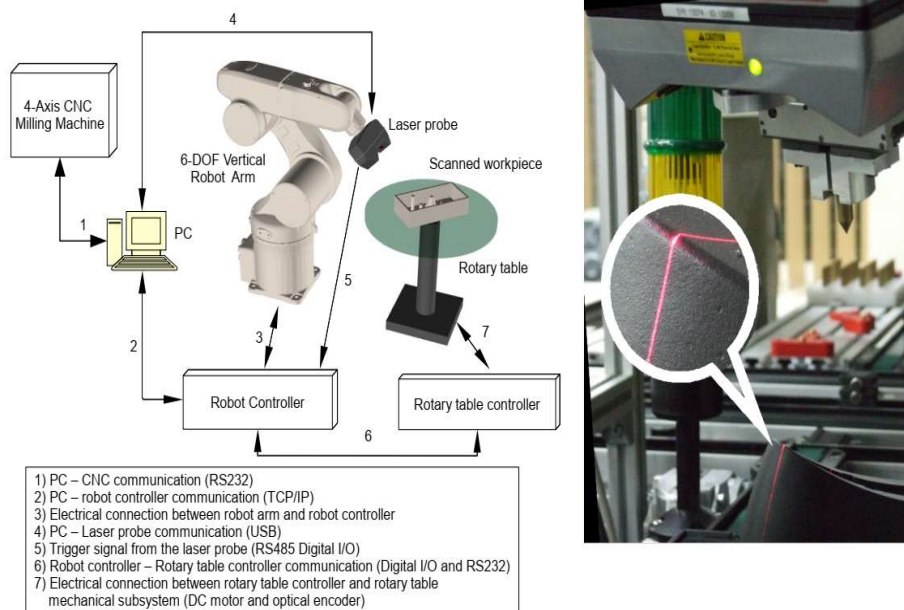


Figure 2.6: Integrated system for 3D reconstruction of objects. [31]

## 2.3 Trajectory planning and optimization

Trajectory planning and optimization requires accurate knowledge of the geometry of the object on which paint is to be deposited. Geometry is usually available in the form of a CAD model or a 3D scan from a sensor device. Once the geometry of the surface is acquired, a spraying process model needs to be established to describe the physics of the coating thickness on the surface. A trajectory for the paint gun can then be generated to cover the entire surface while ensuring paint quality and other objectives. Paint quality can be qualitatively described as the uniformity of coating thickness over a painted surface. This section of literature review analyzes the techniques used for optimizing trajectories over complex free-form surfaces.

### 2.3.1 Trajectory planning with paint quality optimization

An automated trajectory planning scheme is used to find spray trajectories of unknown parts [34]. This method uses a direct *PFeatureDetector* (Process Oriented Feature Detector) based approach to extract elementary geometries from a range sensor data. This is done by first removing the skid, calibrating, and separating the part. After this, for each segmented part, an edge map and mesh are generated. This approach is limited to the detection of two types of geometries: Rib Detection and Cavity Detection. Rib detection detects parallel lines representing ribs while cavity detection finds a region lower than the neighboring regions using surface normal.

An incremental approach for trajectory generation of spray-painting robots is proposed [35]. This method uses several parameters like a surface model, spray and gun model, paint distribution model, spray pattern, and desired coating thickness to generate a spray gun trajectory. The geometry of the part is expressed in the form of triangular patches using a CAD model. To determine the coating thickness over the surface, a circular paint distribution model is employed. Similarly, a functional mapping between the thickness of paint applied and important parameters like spray gun radius, the paint flow rate, and paint

transfer efficiency are obtained. The velocity of the paint gun and the overlap distance are optimized to improve coating distribution over the surface.

Furthermore, the use of Bezier curves to plan spray painting trajectories is investigated [36]. The spraying process is modelled using paint distribution on a circular area, while the overlap is assumed constant over the surface. The use of T-Bezier curves in trajectory planning ensures computational efficiency. The trajectories are planned along the U and V principal directions of the geometry. Results show U direction trajectory gives better coating thickness ( $51.1 \mu\text{m}$ ), and lower process time (82 s) as tabulated in Table 2.1.

Table 2.1: Trajectory optimization results for U and V direction trajectories. [36]

	<b>U direction</b>	<b>V direction</b>
<b>Desired (<math>\mu\text{m}</math>)</b>	50.0	50.0
<b>Average (<math>\mu\text{m}</math>)</b>	51.1	52.2
<b>Maximum (<math>\mu\text{m}</math>)</b>	56.3	58.3
<b>Minimum (<math>\mu\text{m}</math>)</b>	45.2	43.1
<b>Process time (s)</b>	82	99

Recent studies show the use of point cloud slicing technique in conjunction with the coating thickness model to generate paint trajectories [37]. These methods are based on the geometry of the object obtained via a laser sensor. A coating thickness model is established by defining key geometric variables on the free-form surface. Next, a slicing technique is used to obtain a particular portion of the point cloud. A grid projection algorithm is then used to acquire points within the slice for computing coating thickness over them. Finally, a golden section method is used to obtain the optimal slice width, and velocity of the paint gun. This process is repeated for all the slices until the entire surface is covered. The spraying process model is shown in Fig. 2.7.

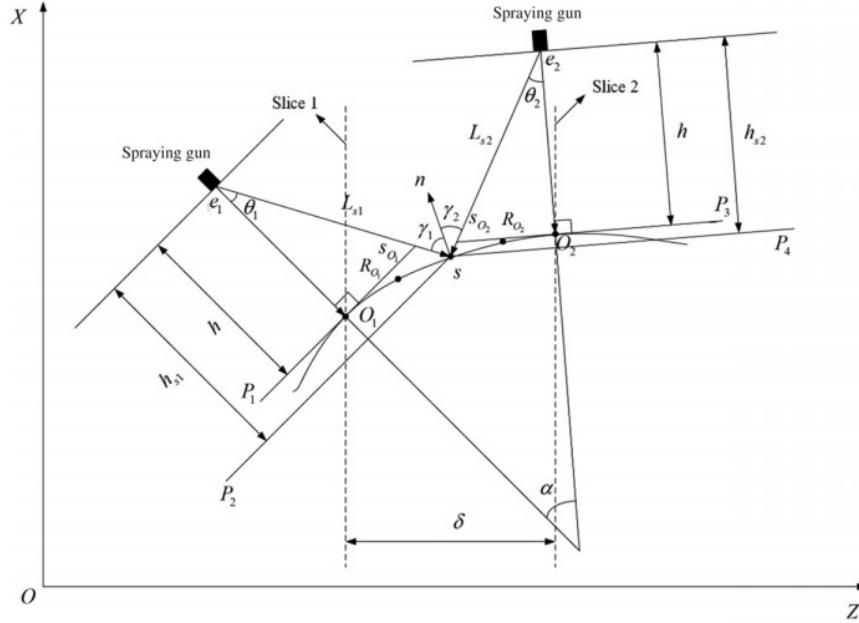


Figure 2.7: Coating thickness model for a complex free-form surface. [37]

The coating thickness is modelled using a double beta distribution. Using equidistant slicing and a desired coating thickness of  $23 \mu m$ , a mean coating thickness of  $25.947 \mu m$  is obtained over the surface of a motorcycle spoiler. The maximum and minimum values are  $34.022 \mu m$  and  $7.928 \mu m$  respectively. On the contrary, the use of non-equidistant slicing scheme improves the mean coating thickness to  $22.2669 \mu m$ . The variation in coating thicknesses is also reduced indicated by the maximal and minimal values of  $29.795 \mu m$  and  $6.971 \mu m$  respectively. The variable overlap distances make the paint distribution more uniform and improve the paint quality.

Another research focuses on optimizing the transitional segment of the trajectory points on complex free-form surfaces [38]. The trajectory planning is based on the geometry of the surface obtained via the STL (Stereo Lithography) file. An STL file is a combination of normal vectors and vertices of the associated triangles of a 3D object. The trajectory planning is done by first introducing the slice planes onto the workpiece and then offsetting the points by  $h$  units along the normal direction. The transitional segments can be straight, convex, or concave and are evaluated for a range of beta angles. It is observed that for smaller beta angles, the straight trajectory works better while for moderate to large

angles, the concave trajectory is better. This holds for both convex and concave-type free-form surfaces. The paint quality metric (error) is the ratio of std. deviation and mean of the coating thickness over the surface. The error is improved from 11.13% to 7.81% when transitional segments are used. Results are shown in Fig. 2.8.

State	Approach	Location	Average paint thickness ( $\mu m$ )	Standard deviation ( $\mu m$ )	Error (%)		
Simulation	Before optimization	Segment 1 at the middle of workpiece	69.1	6.3	9.1		
		Segment 2 at the middle of workpiece	72.3	5.9	8.2		
		Segment on the left side of workpiece	67.4	7.4	11.0		
		Segment on the right side of workpiece	68.4	8.2	12.0		
		Segment on the top side of workpiece	70.7	8.5	12.0		
		Segment on the bottom side of workpiece	68.3	6.9	10.1		
	After optimization	Segment 1 at the middle of workpiece	71.6	2.7	3.8		
		Segment 2 at the middle of workpiece	74.2	3.1	4.2		
		Segment on the left side of workpiece	71.1	5.6	7.9		
		Segment on the right side of workpiece	69.2	6.7	9.7		
		Segment on the top side of workpiece	67.4	5.8	8.6		
		Segment on the bottom side of workpiece	67.8	6.1	9.0		
		Experiment	Before optimization	Segment 1 at the middle of workpiece	68.8	6.7	9.8
				Segment 2 at the middle of workpiece	71.4	5.8	8.1
Segment on the left side of workpiece	66.3			6.8	10.2		
Segment on the right side of workpiece	67.5			9.1	13.5		
Segment on the top side of workpiece	66.8			7.3	10.9		
Segment on the bottom side of workpiece	71.2			10.9	15.3		
After optimization	Segment 1 at the middle of workpiece		68.7	3.2	4.6		
	Segment 2 at the middle of workpiece		69.4	4.7	6.8		
	Segment on the left side of workpiece		67.4	4.9	7.3		
	Segment on the right side of workpiece		69.3	5.8	8.4		
		Segment on the top side of workpiece	68.1	5.9	8.7		
		Segment on the bottom side of workpiece	72.6	7.1	9.8		

Figure 2.8: Coating thickness results for transitional segment opt. [38]

### 2.3.2 Trajectory planning with energy optimization

The energy of robotic manipulators can be optimized to generate efficient paint trajectories while ensuring coating uniformity. The trajectory planning of manipulators is dependent on the task it is performing. While the underlying physics of robot energy consumption are similar, the energy optimization mechanism needs to be established for each task. Energy optimization of robots include topology optimization to eliminate needless densities, selection of optimal path and the use of light wight components, etc. [39]. For the paint process, the selection of optimal path is of interest since the components of the robots cannot be altered due to industry standards. An optimal path is generated for the motion of an industrial ABB robot by investigating the energy consumption of multiple trajectories between two points in space [40]. This method uses an SQP (Sequential Quadratic Programming) type algorithm to optimize end-effector velocities leading to low

energy consumption. The trajectories include a right-angle trajectory, straight-line trajectory, an energy-optimal trajectory, a time-optimal trajectory and a trajectory for pick and place maneuver. It is observed that that the energy optimal path is a curved one in the task space and not the straight line one.

Another study uses an invasive weed optimization (IWO) technique to find energy efficient trajectory for a robot using via points while avoiding obstacles [41]. A cost function is established which penalizes redundant joint rotations, and constraints the joint angles to generate a cubic trajectory for the two revolute joints of a serial manipulator. Similarly, another approach searches for points close to the fly target points that lead to a low energy consumption of the robot [42]. The mechanical energy of the robot is computed through the dynamic model and a branch and bound algorithm is then used to scan for all possible motions to find the energy-optimal trajectory. Results show that the energy consumption for a 6 DOF COMAU Racer robot can be reduced to around 41% as tabulated in Table 2.2.

Table 2.2: Energy consumption results for COMAU Racer robot. [42]

	<b>Pick and Place</b>	<b>Passman</b>
$E_{mecc}$ ( <i>original traj.</i> )	0.285058	0.3861133
$E_{mecc,opt}$ ( <i>opt. traj.</i> )	0.21598 (-24.23%)	0.227371 (-41.11%)
$E_{Tot}$ ( <i>original traj.</i> )	1.260339	1.637855
$E_{Tot,opt}$ ( <i>opt. traj.</i> )	1.026233 (-18.57%)	1.1937 (-27.11%)

## 2.4 Trajectory execution in simulation environment

Simulation software plays an important role in the development of science and technology. They provide a convenient method for testing the system without developing any prototype. Likewise, spray paint simulations can be used to predict the end results of the painting without any significant cost. It is also useful for evaluating the paint quality by measuring the surface area covered by the paint and the uniformity of the paint thickness over the entire surface. A simulation environment usually works with physical objects

represented by a CAD model and a physics engine which enables the software to define the interactions between the objects in the environment. The physics engine contains provision for defining the spray paint methods, spray paint trajectories and dynamics of the physics. To execute the paint trajectory in a simulation environment, it is important to understand important concepts and terminologies. These are explained one by one.

#### **2.4.1 CAD models**

A CAD model stores information about the geometry of an object such as edges, corners, and surfaces, and is of great importance because of the information they carry [43]. In general, CAD models can be divided into two main categories: tessellated and parametric [44]. The tessellated model represents an object by using polygonal meshes described by vertices, edges, and faces. A parametric model stores the geometry of an object by using analytical equations. For instance, a cylinder can be described by two parameters: radius and height. Parametric models are efficient for storing geometries of simple shapes and are not suitable for modeling complex shapes. On the other hand, tessellated models are convenient, but more prone to errors due to approximations.

#### **2.4.2 Spray-painting methods**

Spray-painting methods are techniques used to deposit spray liquids on the surface of work pieces. There are multiple spray-painting techniques including Air Atomized Spray, HVLP (High Volume Low Pressure), Airless Spray, Air Assisted Airless Spray, Heated Spray and Electrostatic Spray painting [45]. The air atomized spray method is the most conventional method and is done by mixing air particles of compressed air with the paint. The compressed air causes the paint to atomize in the form of droplets on the surface. Air-atomized spray painting has great heat transfer capabilities and can be used to cool hot metal surfaces too [46]. An improvement of the air-atomized spray method is the HVLP. The low pressure and high volume of air can mix more efficiently with paint and with the lower impact speed on the surface, the wastage of paint is reduced [47]. Airless spray

systems use a high-pressure paint fluid and a nozzle to deposit paint on the surface of an object. An improved version of the airless spray method is the air-assisted airless spray method. This method leads to an increase in paint efficiency by reducing paint waste. Another method is to heat the paint before applying, reducing its viscosity and hence, less pressure is required to push it out of the nozzle. The heated paint adheres efficiently to the surface thereby, minimizing paint waste [48]. Another method is the use of electrostatic principles to deposit paint over the surface of metals. The surface to be painted is grounded and the paint particles are charged to allow them to adhere to the surface [49]. This method can be used only with metal surfaces, which is its major drawback.

### 2.4.3 Robot simulation software

Robot simulation software provides a user-friendly experience to test paint trajectories and build robot programs. It has a provision of components including a library of CAD models of robots, sample objects for testing trajectories, automatic collision avoidance systems, axes limit features, and post processors for generating robot programs. Some commercial simulators for paint robots include: *RoboDK*, *RobCad paint*, *Delfoi Paint*, *RobotStudio* ® *Paint PowerPac*, *OLP Automatic* and *RoboGuide PaintPRO*.

*RoboDK* is a software developed for offline programming of robots with the provision of simulation tools [50]. It has a user-friendly interface but is not able to generate automated trajectories based on the geometries. It is useful for research and testing purposes. Another simulator is the *RobCad paint* by Siemens which is also based on offline programming [51]. It has features such as paint databases and paint coverage analysis. Its main advantage is the availability of predefined paths which speeds up the development of paint simulations. It also reduces processing times and increases manufacturing quality. The main disadvantage of this software is restricted access to some robot models and libraries. *Delfoi Paint* is another software developed by *Delfoi* which provides capabilities like analysis of paint thickness, precise smooth surface simulation, conveyor tracking, and automatic detection of collision [52]. It also supports post-processors for the



programming of industrial robots. *RobotStudio*® paint *PowerPac* by ABB is yet another commercial simulator that provides the ability to generate robot programs for multiple robots simultaneously [53]. Its main shortcoming is the ability to generate robot trajectories automatically. Some simulation software also provides the ability to generate robot paths. OLP Automatic developed by INROPA is a complete integrated system for automatically generating robot programs [54]. It uses a LASER scanner to scan the workpiece and construct a 3D model of the object, creates a robot path, and then executes it by generating an appropriate robot program for the controller. Furthermore, *RoboGuide PaintPRO* developed by FANUC is another robot simulation software which can generate robot paths automatically [55]. A robot path is generated by simply selecting the surface area of the object to be painted while specifying the correct options. A summary of robot simulation software is given below in Table 2.3.

Table 2.3: Summary of commercial robot simulation software.

	<b>RoboDK</b> [50]	<b>RobCAD Paint</b> [51]	<b>Delfoi paint</b> [52]	<b>RobotStudio Paint Power Pac</b> [53]	<b>OLP Automatic</b> [54]	<b>RoboGuide Paint PRO</b> [55]
<b>Company</b>	RoboDK	Siemens	Delfoi	ABB	INROPA	FANUC
<b>Automatic Trajectory Planning</b>	No	No	No	No	Yes	Yes
<b>Integrated System</b>	No	No	No	No	Yes	No
<b>Paint Thickness Analysis</b>	N/A	N/A	Yes	N/A	N/A	N/A
<b>Post Processors</b>	Yes	Yes	Yes	Yes	Yes	Yes

## 2.5 Trajectory execution on industrial painting robots

Trajectory execution on industrial painting robots refers to the process of converting trajectories (path and velocity commands) to a robot program that can be understood by a robot. Since robot controllers are diverse, each uses its own set of rules to construct the programs. This section briefly describes the robots used for industrial painting processes and the programs used by them for performing trajectories.

### 2.5.1 Industrial painting robots

Apart from the 3D scanning system and trajectory optimization, it is important to consider the experimental setup and validation techniques for the painting process. The experimental setup contains a 3D scanning system, an industrial paint robot with a spray-painting setup, and a robot controller program. Since paint applications have increased considerably due to industrial developments, certain companies have specialized robots for paint applications. A paint robot, unlike other industrial manipulators, usually has less payload capacity, more repeatability, and a hollow wrist for routing the paint lines. A paint robot must also be explosion-proof since the paint material can be inflammable. It should be certified for ATEX EX II certification [56]. Some of the specialized robots are KR AGILUS KR 10 R1100, P-250iB/15, and IRB 5500 Flex Painter. The KR AGILUS KR 10 R1100 is specialized for paint applications by of *KUKA* and *Durr* [57]. It has a wrist payload capacity of 10 Kg, a maximum reach of 1100 mm, and a total of 6 axes as shown in Fig. 2.9. Similarly, the P-250iB/15 developed by FANUC (Fig. 2.10) is one of the largest robots for paint applications [58]. It has a maximum reach of 2800 mm, a load capacity of 15 kg, and a total of 6 axes. It can be attached to walls, and floors, and can also be mounted on rails to reach the desired pose under difficult and narrow regions. Another paint robot developed by ABB named IRB 5500 Flex Painter (Fig. 2.11) is also capable of industrial painting [59]. A summary of robots is tabulated in Table 2.4.



Figure 2.9: KR AGILUS KR 10 R1100. [57]



Figure 2.10: P-250iB/15 by FANUC. [58]



Figure 2.11: IRB 5500 Flex Painter by ABB. [59]

Table 2.4: Industrial robots for paint applications.

Robot Name	Manufacturer	Payload Capacity	Maximum Reach	Number of Axes
<b>KR AGILUS KR 10 R1100</b> [57]	KUKA and Durr	10 kg	1100 mm	6
<b>P-250iB/15</b> [58]	FANUC	15 kg	2800 mm	6
<b>IRB 5500 Flex Painter</b> [59]	ABB	13 kg	2975	6

### 2.5.2 Robot programs

A robot program is a set of instructions to control the movements of a robot. Every robot controller uses its own set of rules for a robot program. A robot program is usually in two formats:

- .LS (List) format
- .TP (Teach Pendant) format.

A List file (.LS) is an ASCII based list file which is not compiled and cannot be executed directly by the robot controller. A .TP file on the other hand is a binary file which is directly executed by a robot controller [60]. To convert a robot trajectory to an LS file, simulation software or a programming language like Python can be used [61]. Simulation software can directly convert the robot trajectory into a robot program by selecting the appropriate post processor. The post processor generates a List file LS that contains robot programs line by line. On the other hand, a custom Python script can also be used to convert the robot path/trajectory directly into LS file. An LS file can then be uploaded directly to a robot controller for execution or can be converted to .TP binary file before upload. For conversion to .TP file, tools like *WinOLPC* can be considered [62].

## 2.6 Validation of paint quality

For the validation of the paint quality, multiple techniques can be used including the calculation of mean and standard deviation of paint thickness on the surface of the work piece, a fractional error metric, propriety coating sensors like ELCOMETER 345, and image processing pipelines. The mean and standard deviation are obtained from the paint deposition model and the accuracy metric outlined by [38]. However, to evaluate the paint thickness model in real time after the paint is applied, image processing can be used [63]. This image processing pipeline starts by taking the image of the painted through a camera. The image is first converted into a binary format using the OPENCV library followed by a noise filtering algorithm [64]. Finally, the image is restored again in terms of original pixel intensities and the distribution of paint is obtained. The stages of image processing pipeline are shown in Fig. 2.12.

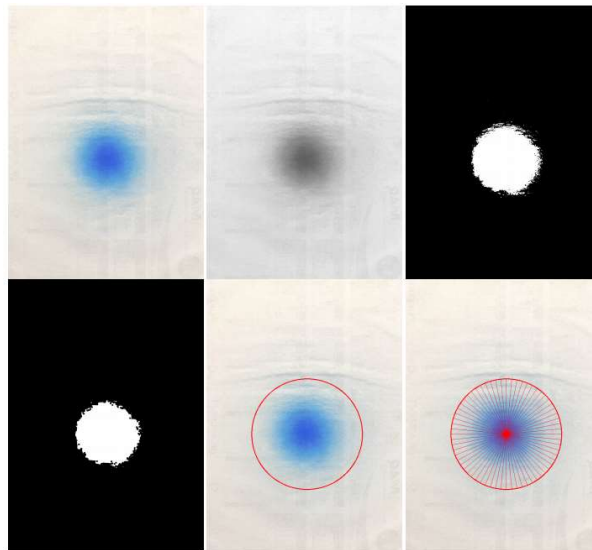


Figure 2.12: Image processing pipeline for paint validation. [63]

An experimental system for the validation of a color spray model is proposed by [10]. It uses a UR-10 robot, a HP-M2 airbrush and a scanner for digitizing the color intensities. The algorithm to convert spray flow rate to the corresponding color intensity starts by defining a variable  $\varphi(r)$ . This function is called Painting Flow function which describes the paint flow rate per unit area. The amount of paint deposited on an incremental

surface area is calculated by transforming the color intensities to paint amounts using a series of conversions. The paint flow function is approximated using gaussian distribution governed by equation 2.1. The radius of the circle is represented by  $r$ , while the parameters A and B are obtained experimentally. It is observed that the distribution of the paint intensity changes by changing the parameters like paint flow rate, the paint gun velocity, and the distance between the spray nozzle and the target surface. The mean coating thickness decreases when the paint process time is decreased and vice versa.

$$\phi(r) = Ae^{-Br^2} \quad (2.1)$$

## 2.7 Research objectives

The problems associated with manual robotic painting can be mitigated using an autonomous robotic system. The literature review highlighted key technologies needed to achieve this task. The primary problem in autonomous painting is trajectory planning based on the geometry of the surface. While much work has been done to optimize the paint quality by searching for an optimal paint trajectory, the optimization of process time and robot energy in conjunction with paint quality is left unattended. Thus, this research aims to:

1. Develop a hybrid optimization scheme to optimize paint quality, process time, and energy consumption of the trajectory planning process by taking into consideration the dynamics of the spraying process and the robot.
2. Design a mechanism to acquire the 3D geometry of the object under investigation for trajectory planning purposes.
3. Design an automated system to perform the experimental analysis on the proposed optimization technique with a user-friendly GUI.

## Chapter 3. Design of 3D Scanning System

### 3.1 Introduction

The literature review highlighted multiple techniques that could be used to acquire a 3D model of an object. These techniques are broadly divided into passive and active methods. Active methods use a light source and is independent of the lighting condition of the space where the object of interest is situated. Passive methods, however, require the surface of the object to be illuminated. To eliminate the need for an active light source outside the hardware of the 3D sensor, the solution to 3D scanning problem boils down to two sensors. One class of sensors uses LASER beams and includes one point, line, and snapshot sensors. These sensors are very expensive and have a very short range typically 200-500 mm. Another class of sensors uses an RGBD sensor which merges an RGB image and a depth image and converts it into a point cloud. These sensors are cheaply available and have a measurement range of 250 mm to 10 m. To capture the depth maps of the object, Intel Real Sense D435 [65] is used. For locating the online position of the axis of rotation, a one-point depth sensor is used [66]. These sensors are shown in Fig. 3.1. The RGB and depth sensors of D435 have a resolution of 1920 x 1080@30 fps, and 1280 x 720@90 fps respectively. Its measurement range is between 0.3 m and 3 m, while the depth accuracy is less than 2% at 2 m. The measurement range of VLX53LoX sensor is 3 cm to 2 m.

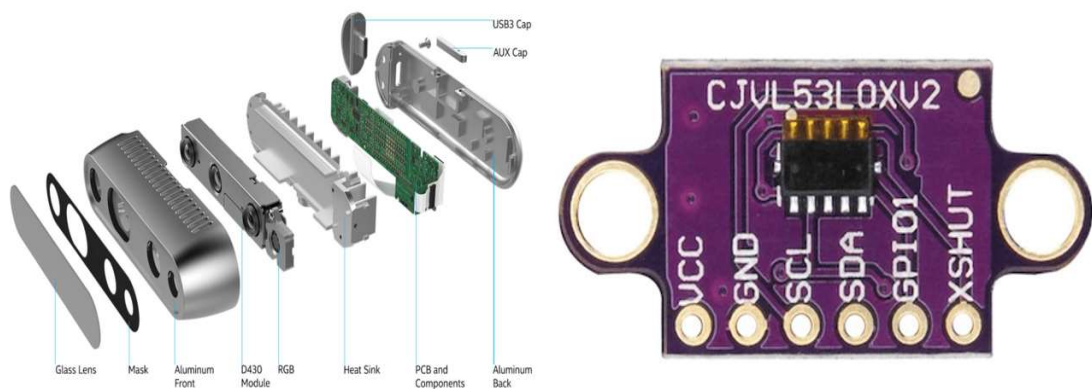


Figure 3.1: (left) Intel Real Sense D435 sensor [65], (right) VLX53LoX [66].

### 3.2 Methodology

To establish a 3D scanning system for generating the geometry of the object, a rotating turntable mechanism is used. A servo motor [67] for the turntable mechanism can be controlled precisely at a resolution of  $1^\circ$ . The D435 sensor [65] is placed at 0.47 m from the object. This allows for the inclusion of thicker objects to be scanned since the minimum range of the sensor is 0.3 m. The object is then rotated and the RGB and depth images are stored for each angular position. An incremental index of  $30^\circ$  is selected to save computational resources. These RGB and depth images are then converted into point clouds using the camera projection matrix [68]. After the point clouds are obtained, a box filter is applied to extract the region of interest. The box filter removes majority of the noisy point cloud data, however statistical noise removal is applied to further refine the point cloud [69]. Finally, raw alignment is applied to align the 3D scans followed by ICP registration to obtain the geometry. A summary of the methodology is shown below in Fig. 3.2.

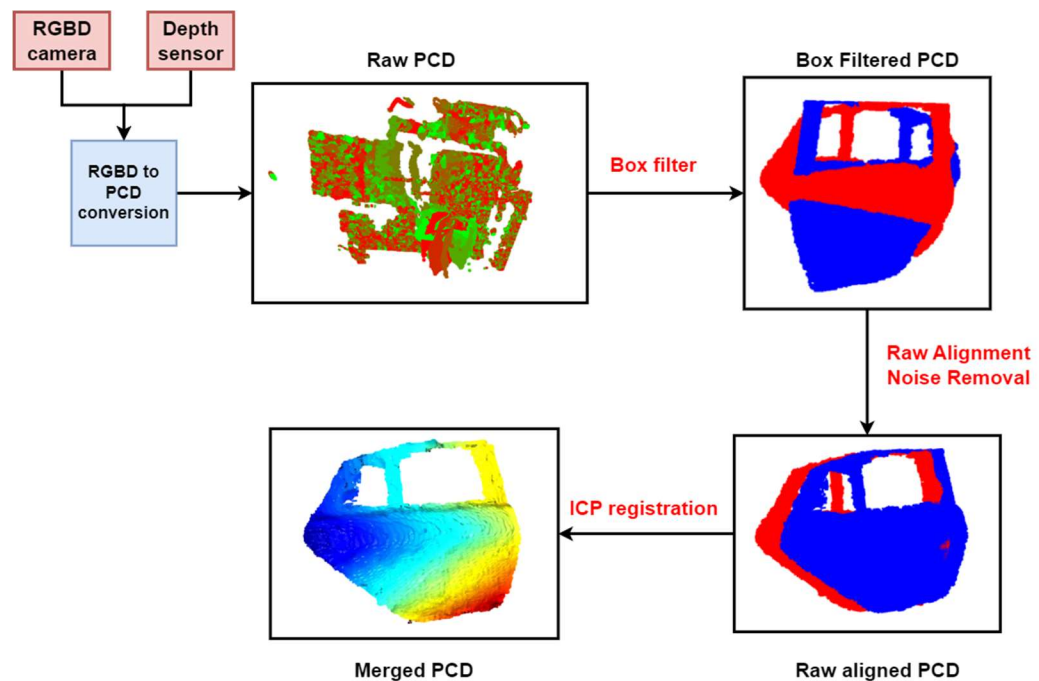


Figure 3.2: Methodology for 3D scan acquisition.



### 3.2.1 Mechanism for 3D scan acquisition

To obtain the complete geometric model of the object, a rotating mechanism is designed as shown in Fig. 3.3. The object of interest is a car door. It is rotated by a servo motor along the axis of rotation. The RGBD sensor is used to capture the depth maps of the object in  $\{C\}$  frame. To estimate the online position of the axis of rotation of the object, a secondary one-point depth sensor is also installed above the main RGBD camera. Its frame of reference  $\{S\}$  is aligned in orientation to the camera frame  $\{C\}$  and is offset by a linear transformation  ${}^C_S T$ . This sensor gives the online transformation matrix  ${}^{S_1} T$  which can be used to raw align the point clouds. Frame  $\{S_1\}$  is the offset frame of reference of the depth sensor aligned with the axis of rotation. Frame  $\{S_{1r}\}$  is the rotated  $\{S_1\}$  frame that rotates along with the object's axis of rotation. The origin of  $\{cg\}$  frame represents the geometric center of the object in  $\{C\}$  frame. The point cloud is termed as  $P^C$  where  $C$  signifies the  $\{C\}$  frame. The object is rotated along its axis of rotation and the RGBD images are stored for each angular position. The two robots' base frames are represented by  $\{0_A\}$  and  $\{0_B\}$  respectively.

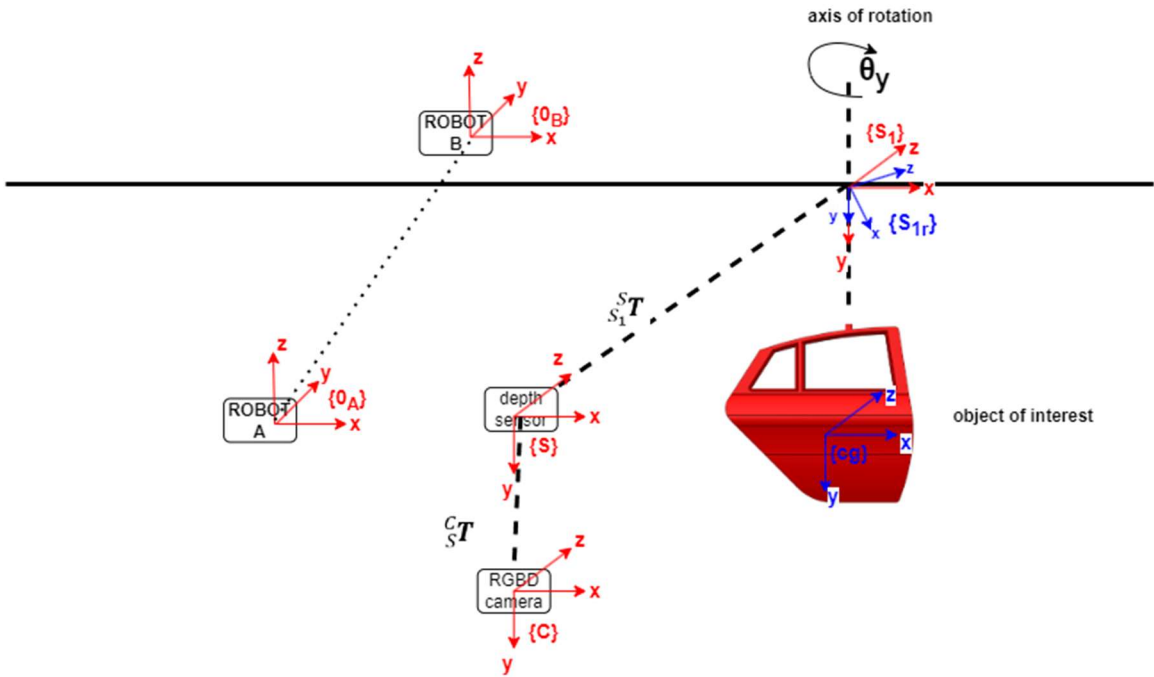


Figure 3.3: Schematic of 3D scan acquisition mechanism.

### 3.2.2 Depth map to point cloud

After the RGBD images are obtained for each angular position, these are then converted into point clouds using the camera projection matrix. To do so, we consider a schematic of the 2D RGB image and the corresponding depth values for each pixel. This is shown in Fig. 3.4.

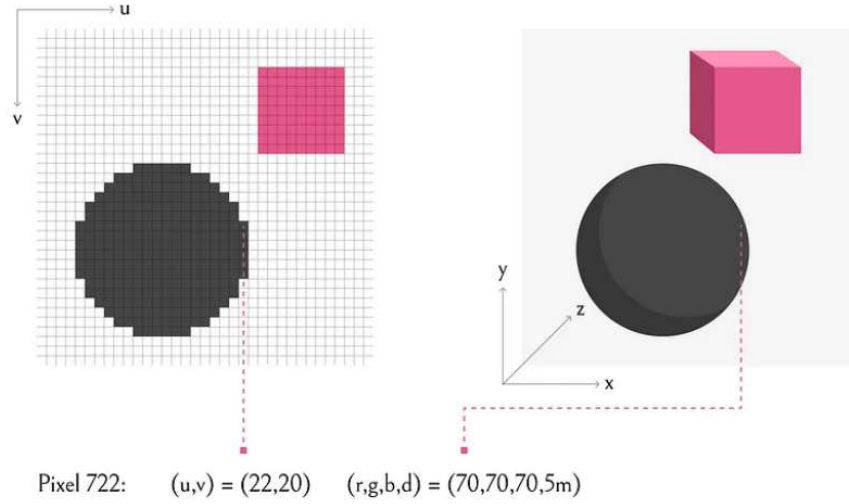


Figure 3.4: Schematic of 2D pixel image and corresponding depth values. [70]

The transformation between the pixel coordinates and cartesian coordinates can be obtained by the application of the camera projection matrix [68].

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = z \begin{bmatrix} \frac{1}{f_x} & -\frac{S}{f_x f_y} & \frac{S c_y - c_x f_y}{f_x f_y} & 0 \\ 0 & \frac{1}{f_y} & -\frac{c_y}{f_y} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \\ -z \end{bmatrix} \quad (3.1)$$

Where  $u$  and  $v$  are the pixel coordinates,  $f_x$  and  $f_y$  are the focal lengths of the camera sensor in x and y direction and  $c_x$  and  $c_y$  are the camera center in the world coordinate.  $S$  is the skew and is zero when the world frame and camera frames are aligned. Using this transformation, the depth images are converted to point clouds.

### 3.2.3 Box Filter to extract region of interest

After the raw point cloud is obtained from the depth fields, the next step is to extract the region of interest. This is done by applying a box filter on the raw point cloud. Since the location of the axis of rotation is known in the  $\{S\}$  frame, it can be used to construct a bounding box around the object. Another reference frame  $\{S_2\}$  is defined at the geometric center of the box such that its origin can be used as an anchor point for the box filter. Using knowledge about the size of the object,  $l_x$ ,  $l_y$  and  $l_z$  can be selected to filter out the object of interest. The box filter is shown in Fig. 3.5.

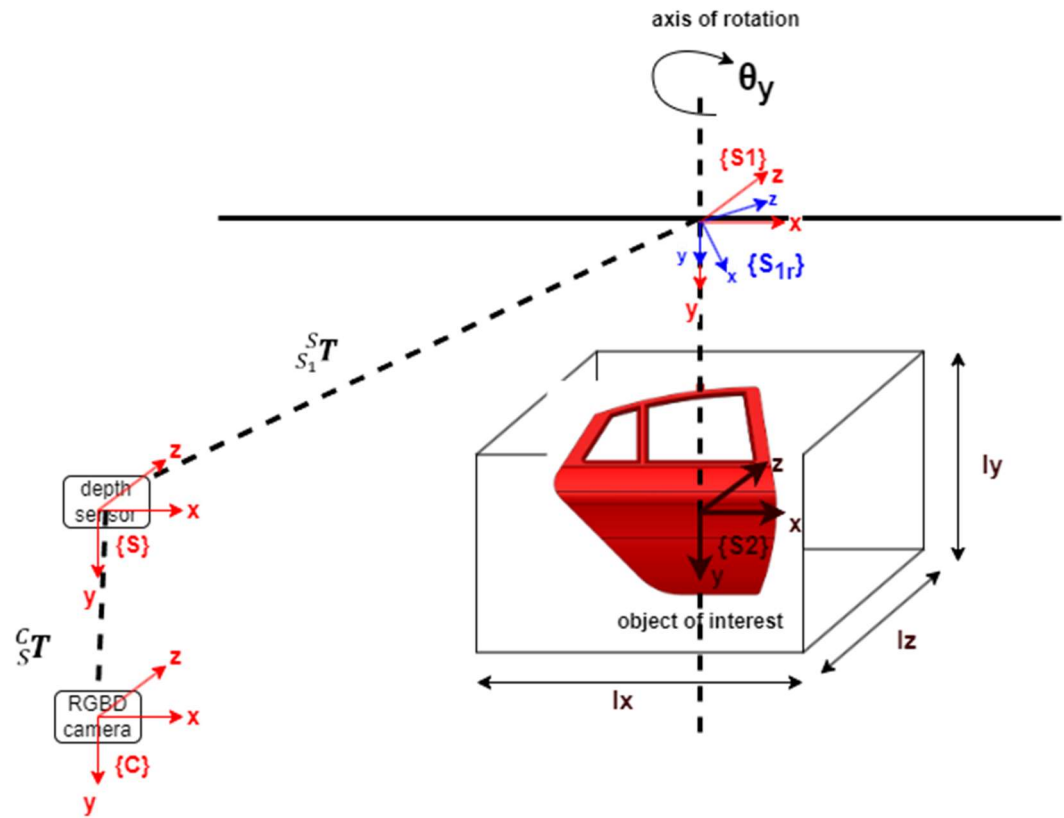


Figure 3.5: Box Filter on raw point cloud.

### 3.2.4 Noise removal and raw alignment of point clouds

Once the region of interest is filtered out, statistical noise removal is applied to remove noisy points and get a clean version of the object. The noise removal is applied using open3d library in Python [69]. A set number of neighbors (700) and a standard deviation ratio of 0.60 is selected for the noise removal. After any residual noise is removed, the point clouds are raw aligned by rotating them backwards along the axis of rotation. Given a point cloud in the camera frame  $\{C\}$ ,  $P^{C(i)}$ , where  $i$  is the rotational index for a rotational angle of  $\theta_{y(i)}$ , the point cloud can be represented in frame  $\{S_{1r}\}$  using the transformations:

$$P^{S_{1r}(i)} = {}^{S_{1r}}_C T P^{C(i)} \quad (3.2)$$

${}^{S_{1r}}_C T$  can be obtained using the following expression:

$${}^{S_{1r}}_C T = {}^{S_{1r}}_{S_1} T {}^{S_1}_S T {}^S_C T \quad (3.3)$$

The transformations  ${}^{S_1}_C T$  and  ${}^S_C T$  can be obtained by using linear offsets along the z and y axes of frame  $\{S_1\}$  and  $\{S\}$  respectively, while  ${}^{S_{1r}}_{S_1} T$  is obtained using a relative rotation matrix along the y-axis such that:

$${}^{S_{1r}}_{S_1} T = R_y(-\theta_{y(i)}) \quad (3.4)$$

The point cloud in the frame  $\{S_{1r}\}$  can be represented by combining equation 3.2, 3.3 and 3.4.

$$P^{S_{1r}(i)} = R_y(-\theta_{y(i)}) {}^{S_{1r}}_{S_1} T {}^S_C T P^{C(i)} \quad (3.5)$$

The aligned point clouds can now be represented in the camera reference frame  $\{C\}$  by translating them back along the z and y axes of frame  $\{S_1\}$  and  $\{S\}$  respectively. This is done by applying the transformation matrix  ${}^C_{S_1} T$ .

$$P^{C(i)}_{aligned} = {}^C_{S_1} T P^{S_{1r}(i)} \quad (3.6)$$

Equation 3.6 represents the raw aligned point cloud in frame  $\{C\}$  for each rotational angle  $\theta_{y(i)}$ . These point clouds can then be registered into one point cloud using ICP.

### 3.2.5 Fine alignment using ICP

After raw alignment, there is still a possibility of misalignment due to measurement errors in the sensor systems. The alignment can be further refined using ICP registration. Both point-to-point and point-to-plane ICP are applied in open3d library [69]. The fitness values of the two are compared and the best one is chosen. The point-to-point ICP aligns a source point cloud  $p_i \in P$  into the reference frame of a source point cloud  $q_i \in Q$  by finding an optimal rotation matrix  $R$  and translational vector  $t$ . If the number of points in the point cloud  $P$  are  $N_{pcd}$ , the point-to-point ICP can be established using the following objective function [27].

$$E(R, t) = \frac{1}{N_{pcd}} \sum_{i=1}^{N_{pcd}} \|q_i - R p_i - t\|^2 \quad (3.7)$$

Similarly, the point-to-plane ICP tries to minimize the following objective function.

$$E(R, t) = \frac{1}{N_{pcd}} \sum_{i=1}^{N_{pcd}} \|(q_i - R p_i - t) \cdot n_{q_i}\|^2 \quad (3.8)$$

$n_{q_i}$  represents the normal vector at point  $q_i \in Q$  in the target space. The optimal rotational matrix ( $R$ ) and translational vector ( $t$ ) are used to align and merge point clouds and get the final scan  $P_{scan}^C$  in the camera reference frame  $\{C\}$ . A schematic of the ICP alignment is shown in Fig. 3.6.

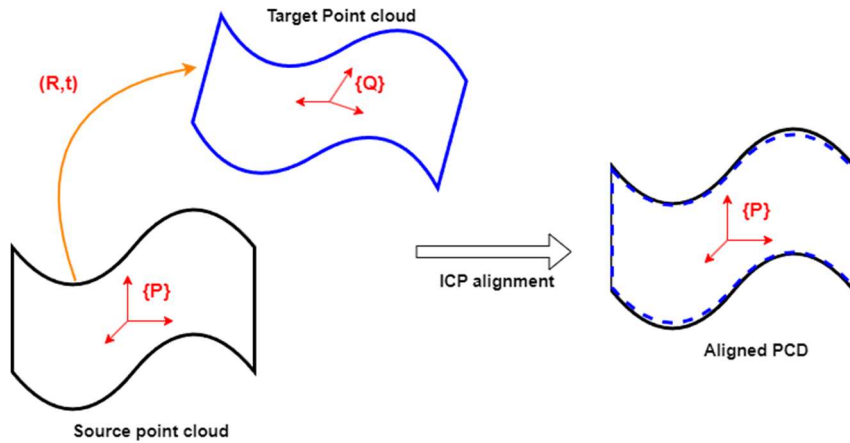


Figure 3.6: Schematic of ICP alignment between source and target point cloud.

### Pseudo code for ICP Fine Alignment

**Input:**  $P_{aligned}^{C(i)}$  (Raw aligned point clouds for each index  $i$ ).  $N = \sum i$  (# of point clouds)

**Output:**  $P_{scan}^C$  (Merged Point cloud in frame  $\{C\}$ )

**Step 1: Assign merged point cloud to the point cloud at index 0.**

$$P_{scan}^C = P_{aligned}^{C(i)}$$

**For  $i$  in range  $(N - 1)$ : (Perform step 2 to step 5)**

**Step 2: Apply Point to point ICP and obtain optimal  $R$  and  $t$ .**

$$(R, t)_{p2p} = p2pICP(P_{scan}^C, P_{aligned}^{C(i+1)})$$

**Step 3: Apply Point to plane ICP and obtain optimal  $R$  and  $t$**

$$(R, t)_{p2plane} = p2planeICP(P_{scan}^C, P_{aligned}^{C(i+1)})$$

**Step 4: Compare fitness scores.**

*if* ( $fitness_{p2p} \geq fitness_{p2plane}$ ):

$$(R, t) = (R, t)_{p2p}$$

*else if* ( $fitness_{p2p} < fitness_{p2plane}$ ):

$$(R, t) = (R, t)_{p2plane}$$

**Step 5: Transform the source point cloud to the target and merge.**

$$P_{scan}^C = merge(P_{scan}^C, T_{(R,t)}P_{aligned}^{C(i+1)})$$

### 3.2.6 CAD calibration in camera frame

The 3D scanned model can be used for trajectory planning and optimization, however, it often has noise and incomplete geometric details. If a CAD model of the object of interest is preset, it can be used for trajectory planning instead. The CAD model, however, needs to be represented in the camera reference frame to ensure the correct spatial location of the object. This can be done by first transforming both the CAD and scan to their eigen coordinate system followed by ICP alignment. Once the correct ICP rotation matrix and translational vectors are obtained, the CAD is then aligned into the scan Eigen reference frame. Finally, the aligned CAD model is translated back to the camera reference frame using the eigenvectors and the principal center of the scanned model. More specifically, given  $P_{scan}^C$  and  $P_{CAD}$  representing the scan point cloud in frame  $\{C\}$  and the CAD point cloud in some arbitrary frame, their transformations into the eigen coordinate system yield:

$$P_{scan}^{eig} = u_{scan}(P_{scan}^C - c_{scan}) \quad (3.9)$$

$$P_{CAD}^{eig} = u_{CAD}(P_{CAD} - c_{CAD}) \quad (3.10)$$

$u_{scan}$  and  $c_{scan}$  represent the eigenvectors and principal center of the scan while  $u_{CAD}$  and  $c_{CAD}$  represent the eigenvectors and principal center of the CAD. Using ICP,  $P_{scan}^{eig}$  and  $P_{CAD}^{eig}$  are aligned via the rotation matrix  $R$  and translational vector  $t$ . This is done using the following transformations:

$$P_{CAD}^{aligned} = R P_{CAD}^{eig} + t \quad (3.11)$$

The aligned CAD model can now be represented in the camera reference by applying the following transformations:

$$P_{CAD}^C = u_{scan}^{-1} P_{CAD}^{aligned} + c_{scan} \quad (3.12)$$

### 3.2.7 Evaluating accuracy of the 3D scan

The Iterative closest point (ICP) aligns the 3D scans which are then merged into a single 3D model of the object under investigation. It is advisable to develop a pipeline that measures the accuracy of the 3D scanning system. Multiple techniques were discussed in the literature review. Some of these techniques use deep neural networks (Deep Shape [71]) to encode the geometric structure of an object into higher dimensional features, while others use simple analytical metrics like  $D_1$ ,  $D_2$ ,  $D_3$ , and  $A_3$  [29]. These features can then be used to find a similarity index between the two geometric models. While neural networks are more accurate than analytical metrics, they require a lot of data to train and are slow to execute. On the other hand, geometric signatures, although not as accurate as neural networks, are easy to compute and can be used to assess the accuracy of the 3D scanning system. Another approach is to use point-to-point Euclidean errors between the CAD and the scan point clouds also termed as ICP error. Thus, using these analytical metrics, a pipeline for evaluating the accuracy of the 3D scanning system is developed as shown in Fig. 3.7. The analytical metrics are used to compute the density distribution plots called geometric signatures along with the ICP error.

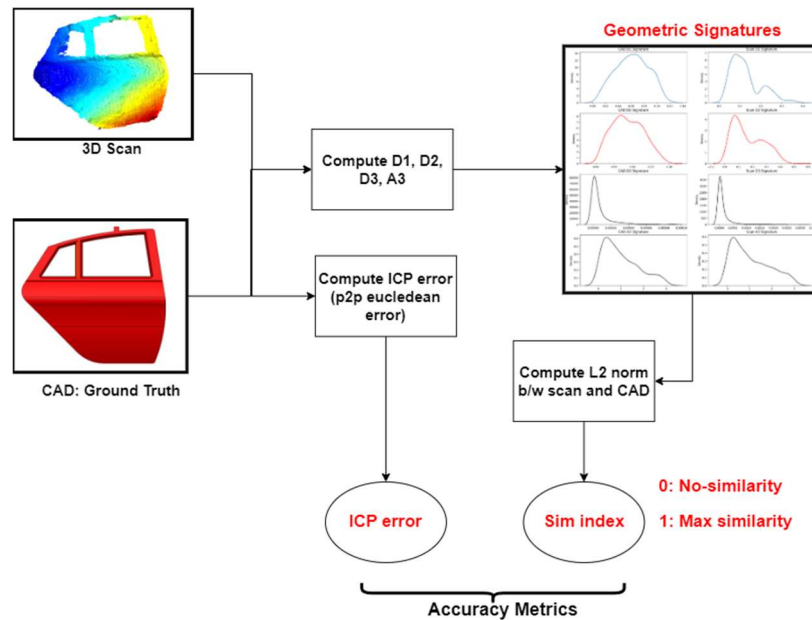


Figure 3.7: Pipeline for evaluating accuracy of the 3D scan.



The analytical metrics for generating shape signatures work by computing distances, areas, and angles between points selected at random in the point cloud. When a sufficiently large sample space of these metrics is obtained, it can capture the underlying structure of the geometry represented on a histogram or a density plot. The shape of the histogram stores information about the underlying structure of the geometry and can be used to evaluate the accuracy of the 3D scanning system by comparing it with the signature of another geometry. For instance,  $D_1$  metric computes the distance of sample points in the point cloud with the geometric center of the point cloud.  $D_2$  computes distance between two random points taken in the point cloud.  $D_3$  computes the area of triangle formed by 3 random points in the point cloud.  $A_3$  computes the angle formed by 3 points in the point cloud. The pseudo codes for calculating these metrics are outlined below.

### 3.2.7.1 Pseudo code for computing $D_1$ metrics

**Input:**  $P$  (point cloud),  $N_{pcd}$  (# of points in point cloud)  $N_{sample}$  (# of sample points)

**Output:**  $D_1$  (An array of size  $N_{sample}$ )

**Step 1: Compute the geometric center of the point cloud:**

$$c = \frac{\sum_{i=1}^{N_{pcd}} P^{(i)}}{N_{pcd}}$$

**Step 2: Generate random indices of size  $N_{sample}$**

$$R_{sample} = rand(N_{sample})$$

**Step 3: Take random sample from the point cloud.**

$$P_{sample} = P[R_{sample}]$$

**Step 4: Compute  $D_1$ :**

$$D_1 = \sqrt{(P_{sample} - c)^2}$$

### 3.2.7.2 Pseudo code for computing $D_2$ metrics

**Input:**  $P$  (point cloud),  $N_{pcd}$  (# of points in point cloud)  $N_{sample}$  (# of sample points)

**Output:**  $D_2$  (An array of size  $N_{sample}$ )

**Step 1: Generate 2 batches of random indices of size  $N_{sample}$**

$$[R_{1sample}, R_{2sample}] = rand(N_{sample})$$

**Step 2: Take two samples from the point cloud.**

$$P_{1samp} = P[R_{1samp}]$$

$$P_{2samp} = P[R_{2sample}]$$

**Step 3: Compute  $D_2$ :**

$$D_2 = \sqrt{(P_{1sample} - P_{2s})^2}$$

### 3.2.7.3 Pseudo code for computing $D_3$ metrics

**Input:**  $P$  (point cloud),  $N_{pcd}$  (# of points in point cloud)  $N_{sample}$  (# of sample points)

**Output:**  $D_3$  (An array of size  $N_{sample}$ )

**Step 1: Generate 3 batches of random indices of size  $N_{sample}$**

$$[R_{1sample}, R_{2samp}, R_{3sample}] = rand(N_{sample})$$

**Step 2: Take three samples from the point cloud.**

$$P_{1samp} = P[R_{1sample}]$$

$$P_{2sample} = P[R_{2sample}]$$

$$P_{3samp} = P[R_{3sample}]$$

**Step 3: Compute side lengths of the triangles:**

$$a = \sqrt{(P_{1sample} - P_{2sample})^2}$$

$$b = \sqrt{(P_{1samp} - P_{3sample})^2}$$

$$c = \sqrt{(P_{2sample} - P_{3sample})^2}$$

$$s = \frac{1}{2}(a + b + c)$$

**Step 4: Compute  $D_3$ :**

$$D_3 = \sqrt{s(s - a)(s - b)(s - c)}$$

### 3.2.7.4 Pseudo code for computing $A_3$ metrics

**Input:**  $P$  (point cloud),  $N_{pcd}$  (# of points in point cloud)  $N_{sample}$  (# of sample points)

**Output:**  $A_3$  (An array of size  $N_{sample}$ )

**Step 1: Generate 3 batches of random indices of size  $N_{sample}$**

$$[R_{1sample}, R_{2sample}, R_{3sample}] = rand(N_{sample})$$

**Step 2: Take three samples from the point cloud.**

$$P_{1sample} = P[R_{1sample}]$$

$$P_{2sample} = P[R_{2sample}]$$

$$P_{3sample} = P[R_{3sample}]$$

**Step 3: Compute vectors from the 3 points with point 1 at the vertex.**

$$u_1 = P_{2sample} - P_{1sample}$$

$$u_2 = P_{3sample} - P_{1sample}$$

**Step 4: Compute  $A_3$**

$$A_3 = \cos^{-1} \left( \frac{\mathbf{u}_1 \cdot \mathbf{u}_2}{|\mathbf{u}_1| |\mathbf{u}_2|} \right)$$

### 3.2.7.5 Pseudo code for evaluating 3D scan accuracy.

**Input:**  $P_{scan}, P_{CAD}$  (Scan and CAD point clouds),  $N_{pcd}$ : Number of points in point cloud and  $N_{sample}$ : Sample points for computing geometric signatures

**Output:**  $S$  (Similarity index)

**Step 1: Compute density distribution for  $D_1$  metrics:**

$$[D_{1scan}, D_{1CAD}] = compute\_D_1(P_{scan}, P_{CAD}, N_{sample})$$

$$S_{D1} = 1 - \sqrt{\frac{\sum (D_{1scan} - D_{1CAD})^2}{N_{sample}}}$$

**Step 2: Compute density distribution for  $D_2$  metrics:**

$$[D_{2scan}, D_{2CAD}] = compute\_D_2(P_{scan}, P_{CAD}, N_{sample})$$

$$S_{D2} = 1 - \sqrt{\frac{\sum(D_{2sca} - D_{2CAD})^2}{N_{sample}}}$$

**Step 3: Compute density distribution for  $D_3$  metrics:**

$$[D_{3scan}, D_{3CAD}] = \text{compute\_}D_3(P_{scan}, P_{CAD}, N_{sample})$$

$$S_{D3} = 1 - \sqrt{\frac{\sum(D_{3scan} - D_{3CAD})^2}{N_{sample}}}$$

**Step 4: Compute density distribution for  $A_3$  metrics:**

$$[A_{3scan}, A_{3CAD}] = \text{compute\_}A_3(P_{scan}, P_{CAD}, N_{sample})$$

$$S_{A3} = 1 - \sqrt{\frac{\sum(A_{3scan} - A_{3CAD})^2}{N_{sample}}}$$

**Step 5: Take mean of the Similarity indices:**

$$S = \frac{S_{D1} + S_{D2} + S_{D3} + S_{A3}}{4}$$

### 3.3 Conclusion

This chapter described in detail the development of a 3D scan acquisition system. A rotating mechanism is employed to allow the exposure of the object to the 3D scanner at multiple rotational indices. An RGBD camera is used to capture the depth maps of the object while a one-point depth sensor is used to estimate the location of the axis of rotation. The depth maps are then converted into point clouds using the camera projection matrix. Once the point clouds are obtained, a box filter is applied to extract the region of interest in the point cloud. To remove any residual noise from the point clouds, a statistical noise removal algorithm is applied. Then, a series of transformations are applied to align the point clouds in the camera reference frame. Further fine alignment is achieved using ICP registration. After an acceptable fitness score is achieved by the ICP registration, the point clouds are then merged into a single point cloud. This point cloud represents the final 3D scan of the object under investigation. Finally, a pipeline for evaluating the accuracy of the 3D scan acquisition system is also developed using the concept of geometric signatures and point to point Euclidean errors. The results and discussion of the 3D scanning system are discussed in detail in Chapter (6).

## Chapter 4. Optimal Paint Trajectory Planning

### 4.1 Introduction

The optimal trajectory planning for paint spraying requires the establishment of the paint spraying model, the coating deposition model, the dynamic model of the robot and the optimization scheme for optimizing the slice thickness, the paint gun velocity, the slicing direction, and the configuration of the robot to achieve uniform paint coating, less process times, and minimal energy consumption. These are described in detail in this chapter.

### 4.2 Methodology

#### 4.2.1 Establishment of spraying process model

The trajectory planning for paint spraying is based on the establishment of the spraying process model. The spraying process model defines how the paint is deposited from the paint gun onto a surface. The spraying area ejecting paint gun is usually in the form of an ellipse and the coating thickness can be modelled using a double beta distribution. Although other models exist, including parabolic distribution, normal distribution and beta distribution, the double beta distribution model has higher accuracy and practicality [37]. Furthermore, it can be converted to a parabolic and a beta distribution by adjusting the values of  $\beta_x$  and  $\beta_y$ . The spraying torch model and the corresponding double beta distribution for the coating thickness are shown in Fig. 4.1 and Fig. 4.2 respectively. The spraying area is elliptical where  $a$  and  $b$  represent the longer and shorter side of the ellipse while,  $\varphi_x$  and  $\varphi_y$  represent the maximum opening angles of the ellipse in the X and Y direction respectively.

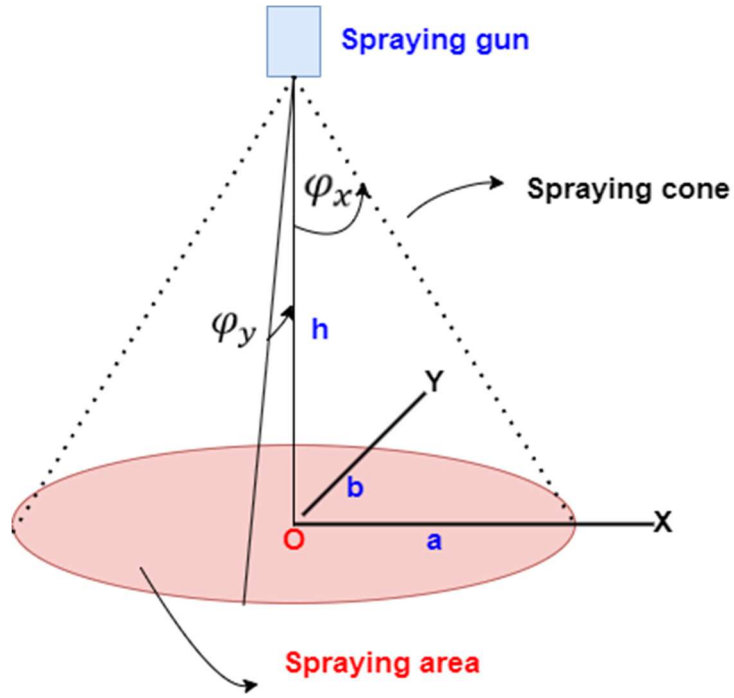


Figure 4.1: Spraying torch model (Elliptical Paint area).

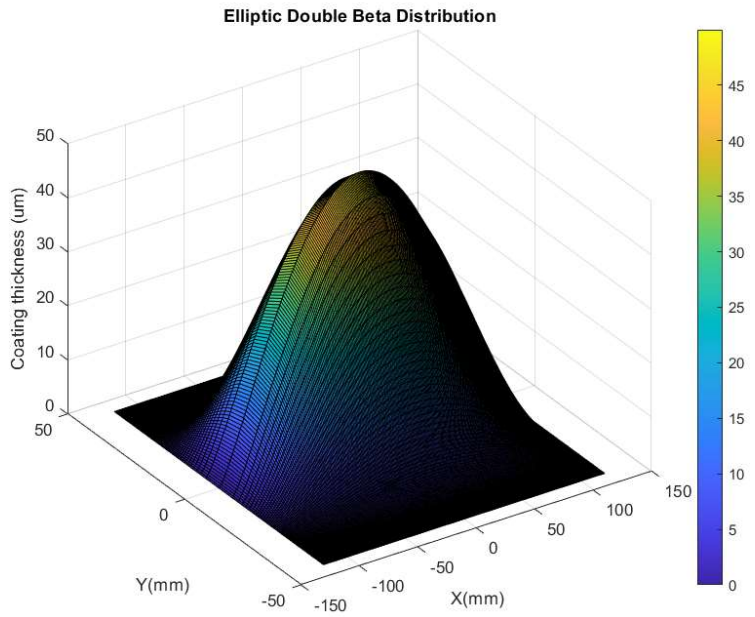


Figure 4.2: Elliptical double beta distribution model of coating thickness on an elliptical surface area.

#### 4.2.2 Establishment of coating deposition model

Once the spraying process model is defined using the double beta distribution, the coating deposition model on a complex free-form surface can now be presented as shown in Fig. 4.3. The point  $s$  represents a point on the free-form surface on which the coating deposition is to be computed. A vertical projection line is constructed from the spraying gun to the free-form surface. A connection line  $L_s$  represents the distance from the spraying gun to the point  $s$  which has a normal vector  $\bar{n}$ . Two tangent planes  $M_1$  and  $M_2$  are drawn at point  $O$  and point  $s$  such that,  $h$  and  $h_s$  represent the vertical height of paint area  $C_1$  and  $C_2$  from the spraying gun. The angle  $\varphi_x$  represents the opening angle of the spray cone along the X direction of the ellipse. Angle  $\gamma$  represents the angle between the normal vector  $\bar{n}$  and the connection line  $L_s$ . Since the coating deposition rate of the spraying gun is conserved, the amount of paint at  $C_1$  and  $C_2$  are equal. The area of paint at  $C_1$  and  $C_2$  are represented by  $A_{C_1}$  and  $A_{C_2}$  respectively. Applying the conservation of paint flow rate, we get:

$$Q_{C_1} = Q_{C_2} \quad (4.1)$$

$$d_{C_1} A_{C_1} = d_{C_2} A_{C_2} \quad (4.2)$$

$$d_{C_2} = \frac{A_{C_1}}{A_{C_2}} d_{C_1} \quad (4.3)$$

The area relationship between point 1 and point 2 can be expressed in terms of  $h$  and  $h_s$  such that:

$$\frac{A_{C_1}}{A_{C_2}} = \left( \frac{h}{h_s} \right)^2 \quad (4.4)$$

Inserting equation 4.4 into equation 4.3 yields:

$$d_{c_2} = \left(\frac{h}{h_s}\right)^2 d_{c_1} \quad (4.5)$$

Accounting for the curvature of the free-form surface, the coating thickness at point  $s$  is then expressed as:

$$d_s = \left(\frac{h}{h_s}\right)^2 \left(\frac{\cos \gamma}{\cos \varphi_x}\right) d_{c_1} \quad (4.6)$$

It should be noted that the coating thickness is zero at point  $s$  if the angle  $\gamma \geq 90^\circ$ . This condition is observed when the free-form surface is exactly vertical at point  $s$ .

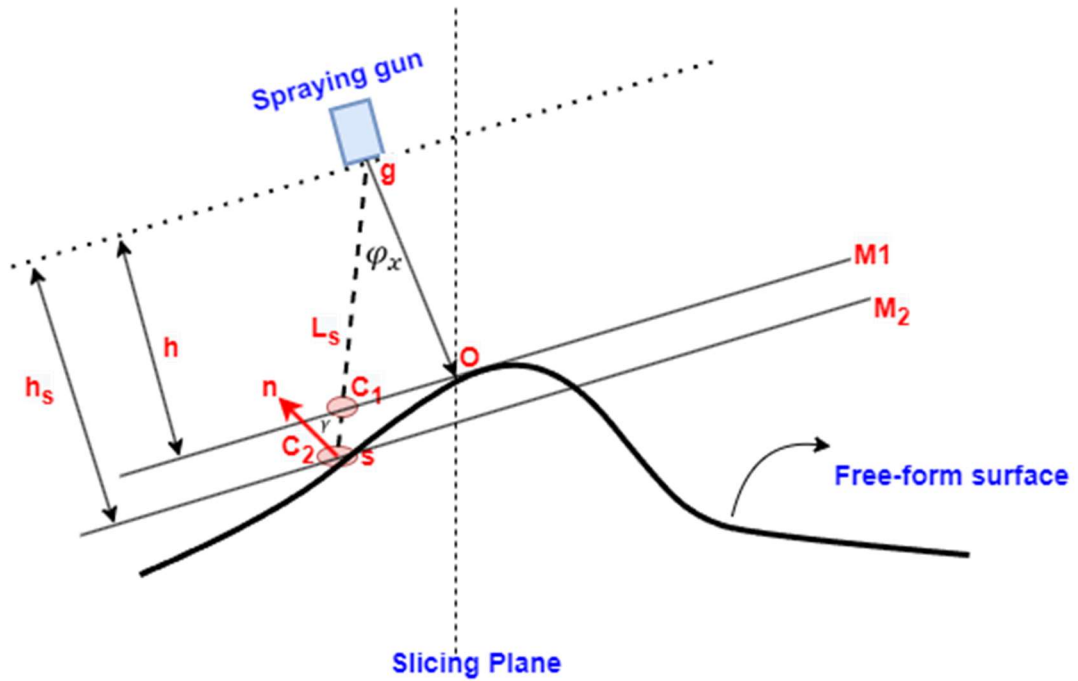


Figure 4.3: Coating deposition model on a complex free-form surface.

The static coating deposition at area  $C_1$  for  $x \in [-a \ a]$  and  $y \in [-b \ b]$  is defined by the equation:

$$d_{c_1}(x, y) = d_{max} \left(1 - \frac{x^2}{a^2}\right)^{\beta_x - 1} \left(1 - \frac{y^2}{b^2 \left(1 - \frac{x^2}{a^2}\right)}\right)^{\beta_y - 1} \quad (4.7)$$



$d_{max}$  represents the maximum static coating thickness deposited at the center of the ellipse. If the spraying gun is moving with speed  $v$  in the Y-direction, the time it takes for the spraying gun to traverse a point  $M (x_M, y_M)$  on a planar surface along the shorter side of the ellipse can be defined by:

$$t_M = \frac{\Delta y}{v} = \frac{2b \left(1 - \frac{x^2}{a^2}\right)^{\frac{1}{2}}}{v} \quad (4.8)$$

Similarly, since the spraying gun is moving with speed  $v$  in the Y-direction, the effective y-coordinate of the ellipse can be formulated as:

$$y_M = b \left(1 - \frac{x_M^2}{a^2}\right)^{\frac{1}{2}} - vt \quad (4.9)$$

The dynamic coating deposition model at  $C_1$  can thus be expressed by putting the values  $y_M$  in the beta distribution model.  $k_{max}$  refers to the dynamic coating thickness (maximum coating thickness per unit time).

$$d_{C_1}(x, y) = \int_0^{t_M} k_{max} \left(1 - \frac{x_M^2}{a^2}\right)^{\beta_x - 1} \left(1 - \frac{\left(b(a^2 - x_M^2)^{\frac{1}{2}} - avt\right)^2}{b^2(a^2 - x_M^2)}\right)^{\beta_y - 1} dt \quad (4.10)$$

The coating thickness at point  $s$  can finally be represented by adjusting equation 4.10 for the curvature of the complex free-form surface.

$$d_s(x, y) = \int_0^{t_M} k_{max} \left(1 - \frac{x_M^2}{a^2}\right)^{\beta_x - 1} \left(1 - \frac{\left(b(a^2 - x_M^2)^{\frac{1}{2}} - avt\right)^2}{b^2(a^2 - x_M^2)}\right)^{\beta_y - 1} \left(\frac{h}{h_s}\right)^2 \left(\frac{\cos \gamma}{\cos \varphi_x}\right) dt \quad (4.11)$$

### 4.2.3 Manipulator forward kinematics model

The coating thickness model on a complex free-form surface governs the amount of coating deposited if the paint gun is moving with a certain velocity. To establish the hybrid optimization scheme, it is important to consider the dynamics of the manipulator performing the paint trajectory. We consider a 4-DOF redundant PRRR manipulator with only position control in the  $x, y, z$  direction. The first joint is a prismatic one for extended reach, while the rest are revolute. The schematic and DH table of the 4-DOF PRRR manipulator are shown in Fig. 4.4 and Table 4.1 respectively.

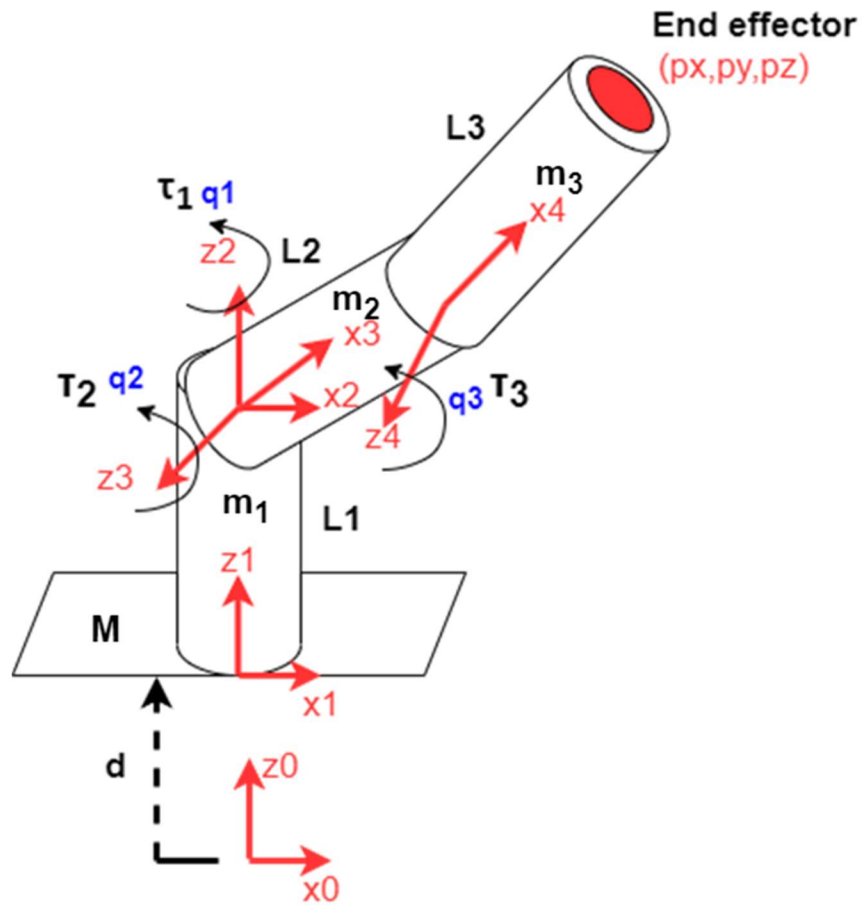


Figure 4.4: DH Schematic of a 4 DOF PRRR manipulator.

Table 4.1: DH Table for 4 DOF PRRR manipulator.

$i$	$\alpha_{i-1}$	$a_{i-1}$	$d_i$	$\theta_i$
1	0	0	$d$	0
2	0	0	$L_1$	$q_1$
3	$90^\circ$	0	0	$q_2$
4	0	$L_2$	0	$q_3$
5	0	$L_3$	0	0

The transformation matrices between consecutive links are:

$${}^0_1T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.12)$$

$${}^2_3T = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_2 & c_2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.13)$$

$${}^3_4T = \begin{bmatrix} c_3 & -s_3 & 0 & L_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.14)$$

$${}^4_5T = \begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.15)$$

Using relative transformations, the transformation matrix between frame {0} and frame {5} can be established as:

$${}^0_5T = {}^0_1T {}^1_2T {}^2_3T {}^3_4T {}^4_5T \quad (4.16)$$

$${}^0_5T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & c_1(L_2c_2 + L_3c_{23}) \\ r_{21} & r_{22} & r_{23} & s_1(L_2c_2 + L_3c_{23}) \\ r_{31} & r_{32} & r_{33} & L_1 + d + L_2s_2 + L_3s_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.17)$$

The end effector position in frame  $\{0\}$  can thus be obtained from the last column of  ${}^0_5T$  matrix:

$$p_x = c_1(L_2c_2 + L_3c_{23}) \quad (4.18)$$

$$p_y = s_1(L_2c_2 + L_3c_{23}) \quad (4.19)$$

$$p_z = L_1 + d + L_2s_2 + L_3s_{23} \quad (4.20)$$

#### 4.2.4 Manipulator inverse kinematics model

The first joint angle is:

$$\mathbf{q}_1 = \mathbf{atan2}(y, x) \quad (4.21)$$

The third joint angle is found using the trigonometric relations:

$$c_3 = \frac{\left(\frac{x}{c_1}\right)^2 + (z - L_1 - d)^2 - (L_2^2 + L_3^2)}{2L_2L_3} \text{ and } s_3 = \pm \sqrt{1 - c_3^2} \quad (4.22)$$

$$\mathbf{q}_3 = \mathbf{atan2}(s_3, c_3) \quad (4.23)$$

For extended reach, variable  $d$  is adjusted. Finally, the second joint angle can be computed using Cramer's Rule.

$$c_2 = \frac{\begin{vmatrix} \left(\frac{x}{c_1}\right) & -L_3s_3 \\ (z - L_1 - d) & L_2 + L_3c_3 \end{vmatrix}}{\begin{vmatrix} L_2 + L_3c_3 & -L_3s_3 \\ L_3s_3 & L_2 + L_3c_3 \end{vmatrix}} \quad (4.24)$$

$$s_2 = \frac{\begin{vmatrix} L_2 + L_3c_3 & \left(\frac{x}{c_1}\right) \\ L_3s_3 & (z - L_1 - d) \end{vmatrix}}{\begin{vmatrix} L_2 + L_3c_3 & -L_3s_3 \\ L_3s_3 & L_2 + L_3c_3 \end{vmatrix}} \quad (4.25)$$

$$\mathbf{q}_2 = \mathbf{atan2}(s_2, c_2) \quad (4.26)$$

#### 4.2.5 Manipulator velocity analysis and Jacobian

The end effector velocity can be equated to the joint space velocity using the Jacobian relationship. This is given by:

$$\dot{x} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = J\dot{q} \quad (4.27)$$

The Jacobian matrix for the 4DOF PRRR configuration can be obtained by finding the derivative of the end effector positions w.r.t joint variables such that:

$$J = \frac{\delta x}{\delta q} = \begin{bmatrix} \frac{\delta x}{\delta d} & \frac{\delta x}{\delta q_1} & \frac{\delta x}{\delta q_2} & \frac{\delta x}{\delta q_3} \\ \frac{\delta y}{\delta d} & \frac{\delta y}{\delta q_1} & \frac{\delta y}{\delta q_2} & \frac{\delta y}{\delta q_3} \\ \frac{\delta z}{\delta d} & \frac{\delta z}{\delta q_1} & \frac{\delta z}{\delta q_2} & \frac{\delta z}{\delta q_3} \end{bmatrix} \quad (4.28)$$

$$J = \begin{bmatrix} 0 & -s_1(L_2c_2 + L_3c_{23}) & -c_1(L_2s_2 + L_3s_{23}) & -c_1L_3s_{23} \\ 0 & c_1(L_2c_2 + L_3c_{23}) & -s_1(L_2s_2 + L_3s_{23}) & -s_1L_3s_{23} \\ 1 & 0 & L_2c_2 + L_3c_{23} & L_3c_{23} \end{bmatrix} \quad (4.29)$$

#### 4.2.6 Manipulator acceleration analysis and Hessian

Hessian matrix relates the task space acceleration vector to joint space acceleration vector governed by:

$$\ddot{x} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = J\ddot{q} + H\dot{q} \quad (4.30)$$

The Hessian matrix is the time derivative of the Jacobian matrix. For the PRRR configuration, it can be defined as:

$$H = \dot{J} = \begin{bmatrix} H_{11} & H_{12} & H_{13} & H_{14} \\ H_{21} & H_{22} & H_{23} & H_{24} \\ H_{31} & H_{32} & H_{33} & H_{34} \end{bmatrix} \quad (4.31)$$

The individual terms of the Hessian matrix are:

$$H_{11} = 0 \quad (4.32)$$

$$H_{12} = -s_1(-L_2s_2\dot{q}_2 - L_3s_{23}(\dot{q}_2 + \dot{q}_3)) - c_1\dot{q}_1(L_2c_2 + L_3c_{23}) \quad (4.33)$$

$$H_{13} = c_1(-L_2c_2\dot{q}_2 - L_3c_{23}(\dot{q}_2 + \dot{q}_3)) - s_1\dot{q}_1(-L_2s_2 - L_3s_{23}) \quad (4.34)$$

$$H_{14} = c_1(-L_3c_{23}(\dot{q}_2 + \dot{q}_3)) - s_1\dot{q}_1(-L_3s_{23}) \quad (4.35)$$

$$H_{21} = 0 \quad (4.36)$$

$$H_{22} = c_1(-L_2s_2\dot{q}_2 - L_3s_{23}(\dot{q}_2 + \dot{q}_3)) - s_1\dot{q}_1(L_2c_2 + L_3c_{23}) \quad (4.37)$$

$$H_{23} = s_1(-L_2c_2\dot{q}_2 - L_3c_{23}(\dot{q}_2 + \dot{q}_3)) + c_1\dot{q}_1(-L_2s_2 - L_3s_{23}) \quad (4.38)$$

$$H_{24} = s_1(-L_3c_{23}(\dot{q}_2 + \dot{q}_3)) + c_1\dot{q}_1(-L_3s_{23}) \quad (4.39)$$

$$H_{31} = 0, \quad H_{32} = 0 \quad (4.40)$$

$$H_{33} = -L_2s_2\dot{q}_2 - L_3s_{23}(\dot{q}_2 + \dot{q}_3) \quad (4.41)$$

$$H_{34} = -L_3s_{23}(\dot{q}_2 + \dot{q}_3) \quad (4.42)$$

#### 4.2.7 Manipulator torque and energy model

The dynamics model of a manipulator in closed form can be written in joint space as:

$$\tau = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + H(q, \dot{q}) \quad (4.43)$$

Where,  $M(q)$  is the inertia matrix dependent on the joint positions,  $C(q, \dot{q})$  is the normal and centrifugal acceleration matrix and is dependent on the joint positions and velocities, while  $H(q, \dot{q})$  contain the gravity and friction dynamics. Gravity is dependent on joint position while the frictional dynamics can be more complex depending on joint positions as well as velocities. In our analysis, the frictional dynamics are excluded to simplify the calculations. The vector  $\tau$  represents the dynamic torque provided to each

joint. The instantaneous power consumption of the manipulator can be found by summing the mechanical power of all the links:

$$P_{mech} = \sum_{n=1}^{N_{joints}} \tau_n \omega_n \quad (4.44)$$

$\tau_n$  is the individual joint torque while  $\omega_n$  is the angular velocity of the corresponding link. The average energy consumed by the manipulator while traversing through point A and B in a time duration of  $t_{AB}$  in the task space can be calculated using:

$$E_{AB} = \frac{(P^{A} + P^{B})}{2} t_{AB} \quad (4.45)$$

#### 4.2.8 Hybrid optimization scheme

Since the coating deposition and the manipulator torque model are established, the hybrid optimization scheme can now be presented. The paint gun is assumed to move along the local y-axis of the ellipse following the curvature of the surface as shown in Fig. 4.5. For each paint stroke, the velocity and the slice thickness must be optimized to achieve a uniform coating thickness on the free-form surface. To optimize a given slice bounded by two slicing planes ( $i$  and  $i + 1$ ), and separated by a distance  $\delta$ , a point of interest ( $s$ ) is considered as shown in Fig. 4.6. The spraying gun is assumed to maintain a constant perpendicular distance  $h$  from the surface. A vertical line is drawn from the spraying gun at the slice plane  $i$  to intersect the surface at point  $O_1$ . A connection line  $L_{s1}$  joins the spraying gun with the point  $s$  making an angle of  $\varphi_{x1}$  with the vertical line. The point  $s$  has a surface normal  $\bar{n}$  defined by the curvature of the locality. The normal vector  $\bar{n}$  makes an angle  $\gamma_1$  with  $L_{s1}$ . The effective x coordinate of the elliptical paint area is  $x_1$ , joining point  $O_1$  and  $s$ . In a similar way, these geometric variables are defined for the slice  $i + 1$ .  $h_{s1}$  and  $h_{s2}$  represent the perpendicular distance of the paint gun with plane  $M_2$  and  $M_4$  respectively. The slicing is performed at an angle  $\theta$  with respect to the eigen coordinate system and the trajectory is planned in this rotated eigen coordinate system called the slicing frame  $\{SF\}$ . The trajectory planning model is shown in Fig. 4.5 and Fig. 4.6 respectively.

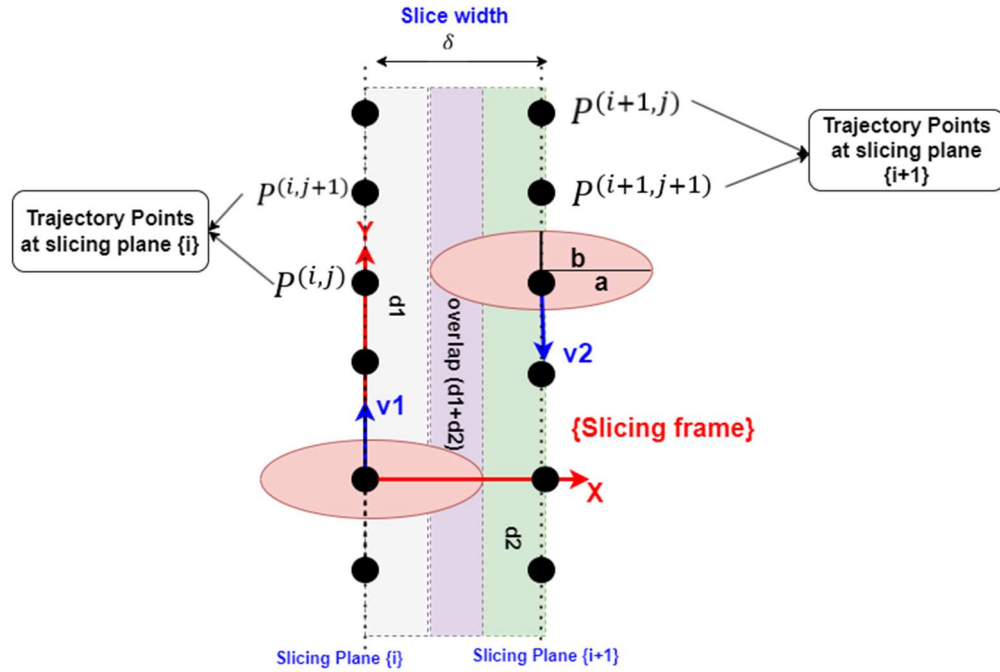


Figure 4.5: Slicing model showing the sliced region, the elliptical paint area, and the trajectory points.

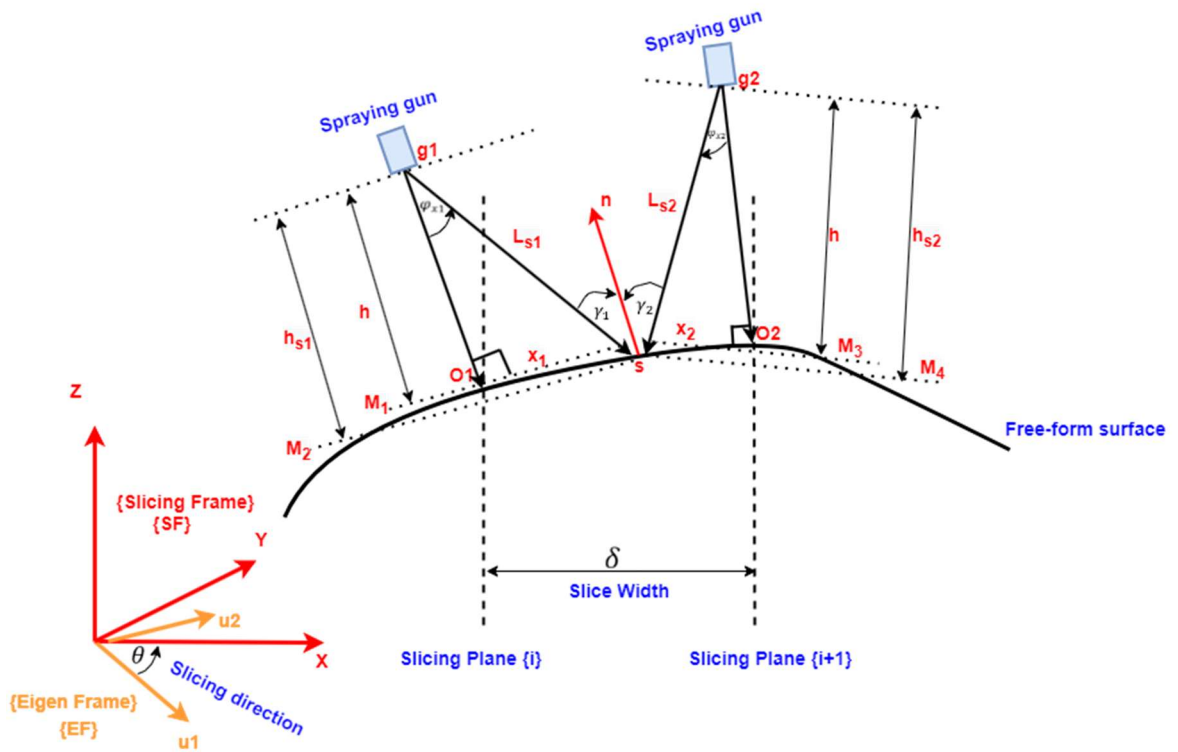


Figure 4.6: Trajectory planning and coating deposition model on a complex free-form surface with slice sandwiched between two spraying gun positions.



Using the variables defined in Fig. 4.6, the coating thickness function at an arbitrary slice plane  $i$  can be presented as:

$$d_i(x, y) = \int_0^{t_i} k_{max} \left(1 - \frac{x_{(i)}^2}{a^2}\right)^{\beta_x - 1} \left(1 - \frac{\left(b(a^2 - x_{(i)}^2)^{\frac{1}{2}} - av_{(i)}t\right)^2}{b^2(a^2 - x_{(i)}^2)}\right)^{\beta_y - 1} \left(\frac{h}{h_{s(i)}}\right)^2 \left(\frac{\cos \gamma_{(i)}}{\cos \varphi_{x(i)}}\right) dt \quad (4.46)$$

The coating thickness at point  $s$  can be  $d_1$ ,  $d_2$  or  $d_1 + d_2$  subject to the overlap conditions. The overlapping conditions are derived by checking the ellipse opening angles  $\varphi_{x1}$ ,  $\varphi_{x2}$  and the angles,  $\gamma_1$  and  $\gamma_2$ . These conditions are outlined below:

$$d_s = \begin{cases} d_1 & \text{if } \varphi_{x1} < \varphi_x^{(max)}, \gamma_1 < 90^\circ, \varphi_{x2} \geq \varphi_x^{(max)} \text{ or } (\gamma_2 \geq 90^\circ) \\ d_1 + d_2 & \text{if } \varphi_{x1} < \varphi_x^{(max)}, \gamma_1 < 90^\circ, \varphi_{x2} < \varphi_x^{(max)} \text{ and } (\gamma_2 < 90^\circ) \\ d_2 & \text{if } \varphi_{x2} < \varphi_x^{(max)}, \gamma_2 < 90^\circ, \varphi_{x1} \geq \varphi_x^{(max)} \text{ or } (\gamma_1 \geq 90^\circ) \end{cases} \quad (4.47)$$

Here,  $\varphi_x^{(max)}$  represents the maximum opening angle of the ellipse along the X direction (longer side) and is computed using the relation:

$$\varphi_x^{(max)} = \tan^{-1}\left(\frac{h}{a}\right) \quad (4.48)$$

The point cloud is sliced at an angle  $\theta$  w.r.t the principal z-direction of the eigen coordinate frame. The paint gun moves along the immediate y-axis of the ellipse following the curvature of the free-form surface to complete a paint stroke. The speed  $v_{(i)}$  for a given slice  $i$  is assumed constant. The individual slices are then sub-divided into patches with each  $b$  distance apart along the y-axis of frame {SF}. The trajectory point for a patch is obtained by displacing the mean position along the normal vector of the patch by  $h$  units. The number of trajectory points for a slice can be increased by decreasing the vertical distance between the patches. Given patch points  $P_{patch} \in \mathbb{R}^{(3, N_{patch})}$  in a portion of slice and the corresponding trajectory point  $P^{(i,j)} \in \mathbb{R}^{(3,1)}$ , the  $L_{s(i)} \in \mathbb{R}^{(3, N_{patch})}$  vector for a can be computed using:

$$\overline{L_{s(i)}} = P_{patch} - P^{(i,j)} \quad (4.49)$$

$N_{patch}$  represents the total number of points in a patch. Using dot product between  $-\overline{\mathbf{L}_{s(i)}}$  and the normal vector  $\overline{\mathbf{n}}$ , angle  $\gamma_{(i)} \in \mathbb{R}^{(1, N_{patch})}$  can be computed as:

$$\cos \gamma_{(i)} = \frac{-\overline{\mathbf{L}_{s(i)}} \cdot \overline{\mathbf{n}}}{|\overline{\mathbf{L}_{s(i)}}| |\overline{\mathbf{n}}|} \rightarrow \gamma_{(i)} = \cos^{-1} \frac{-\overline{\mathbf{L}_{s(i)}} \cdot \overline{\mathbf{n}}}{|\overline{\mathbf{L}_{s(i)}}| |\overline{\mathbf{n}}|} \quad (4.50)$$

Similarly, taking the dot product of  $\overline{\mathbf{L}_{s(i)}}$  and vector  $\overline{\mathbf{h}}$  the angle,  $\varphi_{x(i)} \in \mathbb{R}^{(1, N_{patch})}$  is computed:

$$\cos \varphi_{x(i)} = \frac{\overline{\mathbf{L}_{s(i)}} \cdot \overline{\mathbf{h}}}{|\overline{\mathbf{L}_{s(i)}}| |\overline{\mathbf{h}}|} \rightarrow \varphi_{x(i)} = \cos^{-1} \frac{\overline{\mathbf{L}_{s(i)}} \cdot \overline{\mathbf{h}}}{|\overline{\mathbf{L}_{s(i)}}| |\overline{\mathbf{h}}|} \quad (4.51)$$

Using  $\varphi_{x(i)}$ , the term  $x_{(i)} \in \mathbb{R}^{(1, N_{patch})}$  is calculated:

$$x_{(i)} = h \tan (\varphi_{x(i)}) \quad (4.52)$$

The ratio  $\frac{h}{h_{s(i)}} \in \mathbb{R}^{(1, N_{patch})}$  can be computed using the law of similar triangles such that:

$$\frac{h}{h_{s(i)}} = \frac{\sqrt{x_{(i)}^2 + h^2}}{|\overline{\mathbf{L}_{s(i)}}|} \quad (4.53)$$

Finally, the time duration  $t_{(i)} \in \mathbb{R}^{(1, N_{patch})}$  for each patch point is represented using the relationship:

$$t_{(i)} = \frac{2b \sqrt{\left(1 - \frac{x_{(i)}^2}{a^2}\right)}}{v_{(i)}} \quad (4.54)$$

Once the essential coating parameters are calculated, the coating thickness can be computed as defined by equations 4.46 and 4.47. Next, to compute the dynamic torque of the robot, the velocity vector of the end effector is obtained by finding a unit vector in the direction of delta of two trajectory points. More specifically, given two consecutive trajectory points  $P^{(i,j)}$  and  $P^{(i,j+1)}$  in the slicing frame  $\{SF\}$  along a paint stroke, the velocity vector at a given trajectory point  $j$  within a slice plane  $i$  can be defined as:

$$\bar{v}^{(i,j)} = \frac{(P^{(i,j+1)} - P^{(i,j)})}{\|P^{(i,j+1)} - P^{(i,j)}\|} |v_{(i)}| \quad (4.55)$$

The trajectory points and the velocity vector in the robot reference frame can be found using the relative transformation matrices between the slicing frame {SF}, eigen frame {EF} and the robot base frame {0}.

$$P^{robot} = {}_0^S T P = {}_E^S T {}_S^E T P \quad (4.56)$$

$$V^{robot} = {}_0^S R V = {}_E^S R {}_S^E R V \quad (4.57)$$

The first three rows of matrix  $P \in \mathbb{R}^{(4, N_t)}$  and  $V \in \mathbb{R}^{(3, N_t)}$  represent the trajectory points  $(p_x, p_y, p_z)$  and velocity vectors  $(v_x, v_y, v_z)$ . Similarly, the orientation  $\psi^{(i,j)}$  of the end-effector (i.e., paint gun) at a slicing plane  $i$  and trajectory point  $j$  can be computed by reversing the direction of normal vector that joins point  $O^{(i,j)}$  and  $g^{(i,j)}$ . The orientation matrix  $\psi \in \mathbb{R}^{(3, N_t)}$  can be transformed into the robot frame using the rotation matrix  ${}_0^S R$ .  $N_t$  represents the total number of trajectory points in a slice.

$$\psi^{(i,j)} = -\overline{n_{og}} = \bar{h} \quad (4.58)$$

$$\psi^{robot} = {}_0^S R \psi = {}_E^S R {}_S^E R \psi \quad (4.59)$$

The trajectory points and the velocity vector in the task space are converted to joint space positions and velocities using the inverse kinematics and Jacobian defined earlier. The task space acceleration ( $\ddot{x}$ ) is assumed zero to simplify the calculations. The RNE (Recursive Newton Euler) approach [72] is used to compute the link angular velocities and joint torques to compute the mechanical energy consumption and time duration between trajectory points. A GA (Genetic Algorithm) [73] is then used to optimize the hybrid cost function for a given slice direction ( $\theta$ ), slice width ( $\delta$ ), speeds  $(v_1, v_2)$ , and the inverse kinematic configuration of the manipulator  $ik_{cf}$ . The cost function for the coating thickness is computed by taking the mean squared error of the coating thickness with the ideal coating thickness over  $N_{pts}$  points in the slice.

$$J_{d_s} = \frac{1}{N_{pts}} \sum_{k=1}^{N_{pts}} (d_s^{(k)} - d_{ideal})^2 \quad (4.60)$$

To minimize the coating deviation error, we penalize the ratio of coating thickness standard deviation and mean over a slice.

$$J_{d_{error}} = \frac{d_{std}}{d_{mean}} \quad (4.61)$$

The mean squared error (eq. 4.60) ensures the coating thickness is close to the desired value, while the deviation error (eq. 4.61) ensures uniformity. The standard deviation and mean of the coating thickness over a slice can be computed using the following:

$$d_{mean} = \frac{1}{N_{pts}} \sum_{k=1}^{N_{pts}} d_s^{(k)} \quad (4.62)$$

$$d_{std} = \sqrt{\frac{\sum_{k=1}^{N_{pts}} (d_s^{(k)} - d_{mean})^2}{N_{pts}}} \quad (4.63)$$

Similarly, to minimize the energy consumption of the manipulator, the following objective function is introduced with  $\Delta T^{(n_t)}$  representing the time interval between two consecutive trajectory points:

$$J_E = \frac{1}{N_t - 1} \sum_{n_t=1}^{N_t-1} \left( \left( \sum_{n=1}^{N_{joints}} \frac{1}{2} (\tau_n^{(n_t)} \omega_n^{(n_t)} + \tau_n^{(n_t+1)} \omega_n^{(n_t+1)}) \right) \Delta T^{(n_t)} \right) \quad (4.64)$$

Finally, to ensure fast trajectories, low velocities can be avoided by penalizing the average time between two consecutive trajectory points.

$$J_T = \frac{1}{N_t - 1} \sum_{n_t=1}^{N_t-1} \Delta T^{(n_t)} \quad (4.65)$$

The hybrid cost function is the weighted sum of the four cost functions defined in equation 4.60, 4.61, 4.64, and 4.65. The scaling factors  $\omega_1$ ,  $\omega_2$ ,  $\omega_3$ , and  $\omega_4$  are adjusted to indicate relative importance of penalty functions. It should be noted that the cost functions are normalized on a scale of 0 and 1 before computing the hybrid cost function.

$$J_{tot} = \omega_1 J_{d_s}^{norm} + \omega_2 J_{d_{error}}^{norm} + \omega_3 J_E^{norm} + \omega_4 J_T^{norm} \quad (4.66)$$

The fitness function for the genetic algorithm is then defined as inverse of the cost function and by introducing an  $\epsilon$  term to avoid division by zero.

$$fitness = \frac{1}{J_{tot} + \epsilon} \quad (4.67)$$

The constraints for the optimization objective are:

$$\delta \in [a \quad 2a] \quad (4.68)$$

$$v_1, v_2 \in [v_{min} \quad v_{max}] \quad (4.69)$$

$$ik_{cf} \in \{0 \text{ or } 1\} \quad (4.70)$$

$$\theta \in [0 \quad \pi] \quad (4.71)$$

A schematic of the optimization algorithm is shown in Fig. 4.7. The trajectories once optimized, are analyzed based on four criteria including coating distribution error, relative coating error, energy consumption, and total trajectory time. The relative error is computed by taking the fractional difference of the mean and desired coating thickness over the entire surface.

$$J_{d_{rel}} = \frac{|d_{mean} - d_{ideal}|}{d_{ideal}} \quad (4.72)$$

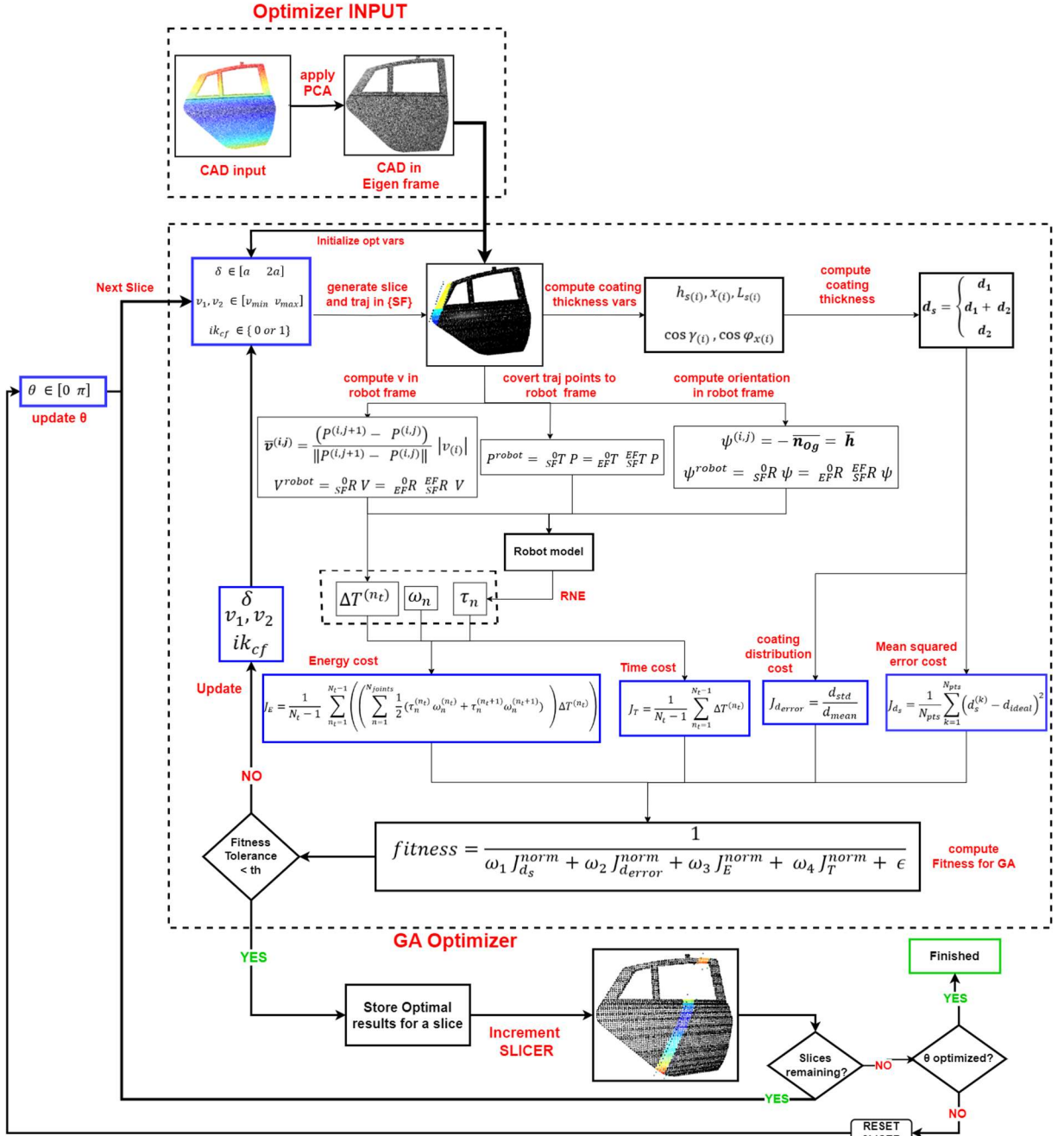


Figure 4.7: Trajectory planning and optimization algorithm. The input to the optimization algorithm is a CAD model while the output is an optimized trajectory for the paint robot in task space. The end-effector trajectory includes the x, y, z location, orientation, and the velocity vector at a given point in task space.

### 4.3 Conclusion

This chapter discussed the detailed process of optimum trajectory planning for a robotic painting process. An elliptic double beta distribution model is employed to model the spraying process. A coating deposition model for a complex free-form surface is then established based on spraying parameters including: the spray gun height from the surface ( $h$ ), the coating deposition rate of the paint per unit time ( $k_{max}$ ), the width and length of the elliptical paint area ( $a, b$ ), the beta values for the paint distribution ( $\beta_x, \beta_y$ ), and the curvature ( $\gamma, \varphi_x$ ). Once the coating deposition model on a complex free-form surface is established, the manipulator kinematic and dynamic model is analytically derived to compute the instantaneous joint torques and link velocities at a given trajectory point. A point cloud slicing algorithm is then used to slice a point cloud of the complex free-form surface for optimization. The trajectory points are assigned along the slicing plane following the curvature of the surface while maintaining a constant height ( $h$ ) from the surface. The coating thickness is then computed for all the points in the given slice, while the manipulator energy is computed for all the trajectory points within this slice. A hybrid cost function is then established to penalize the mean squared error of the coating thickness, the coating distribution error, the mean energy consumption, and the mean trajectory time for one slice. A GA (genetic algorithm) then minimizes this cost function to achieve the optimal slice width ( $\delta$ ), slice velocities ( $v_1, v_2$ ), the slicing direction ( $\theta$ ) and the inverse kinematic configuration ( $ik_{cf}$ ) for all the slices in a point cloud.

## **Chapter 5. Integrated System Development**

### **5.1 Introduction**

The practical implementation of the 3D scanning system and the trajectory optimizer can be realized by developing an integrated system containing components necessary for the automation process. The integrated system consists of hardware and software that work together to make an autonomous system for robotic painting. The hardware components include sensors and actuators for sensing and controlling the various functions of the system. The software allows for smooth integration with the system components. This chapter outlines in detail the development process of the integrated system, its core hardware components and the software used. It also discusses in detail the development of a graphical user interface (GUI) for interacting with the system. The high-level representation of the GUI makes it possible for the system to act autonomously with minimal user intervention. This is made possible by performing the programming logic at the backend while allowing the user to choose macro tasks in the GUI without understanding the details behind it. The GUI has functions for instantiating 3D scanning sequence, visualizing, and validating the accuracy of the 3D scan, running the optimizer for trajectory planning, visualizing the trajectory and uploading the trajectories to the robots while also providing a platform for observing critical sensor readings and camera feeds.

### **5.2 Methodology**

The methodology provides a macro level overview of the hardware and software components needed to develop the autonomous system for robotic painting. The autonomous system is broken down into 4 sub-systems. The 3D scanning system contains hardware and software responsible for generating a 3D model of the complex free-form surface of an object. The trajectory planning system is responsible for generating an optimal paint trajectory. The trajectory execution system enables the system to execute the



trajectory via the robots installed on board. Finally, the validation system is used to validate the 3D scan accuracy of the 3D scanning system, the paint quality and energy consumption of the trajectory planning process. These systems are interconnected and share the hardware and software components to achieve the desired purpose. The methodology is shown schematically in Fig 5.1.

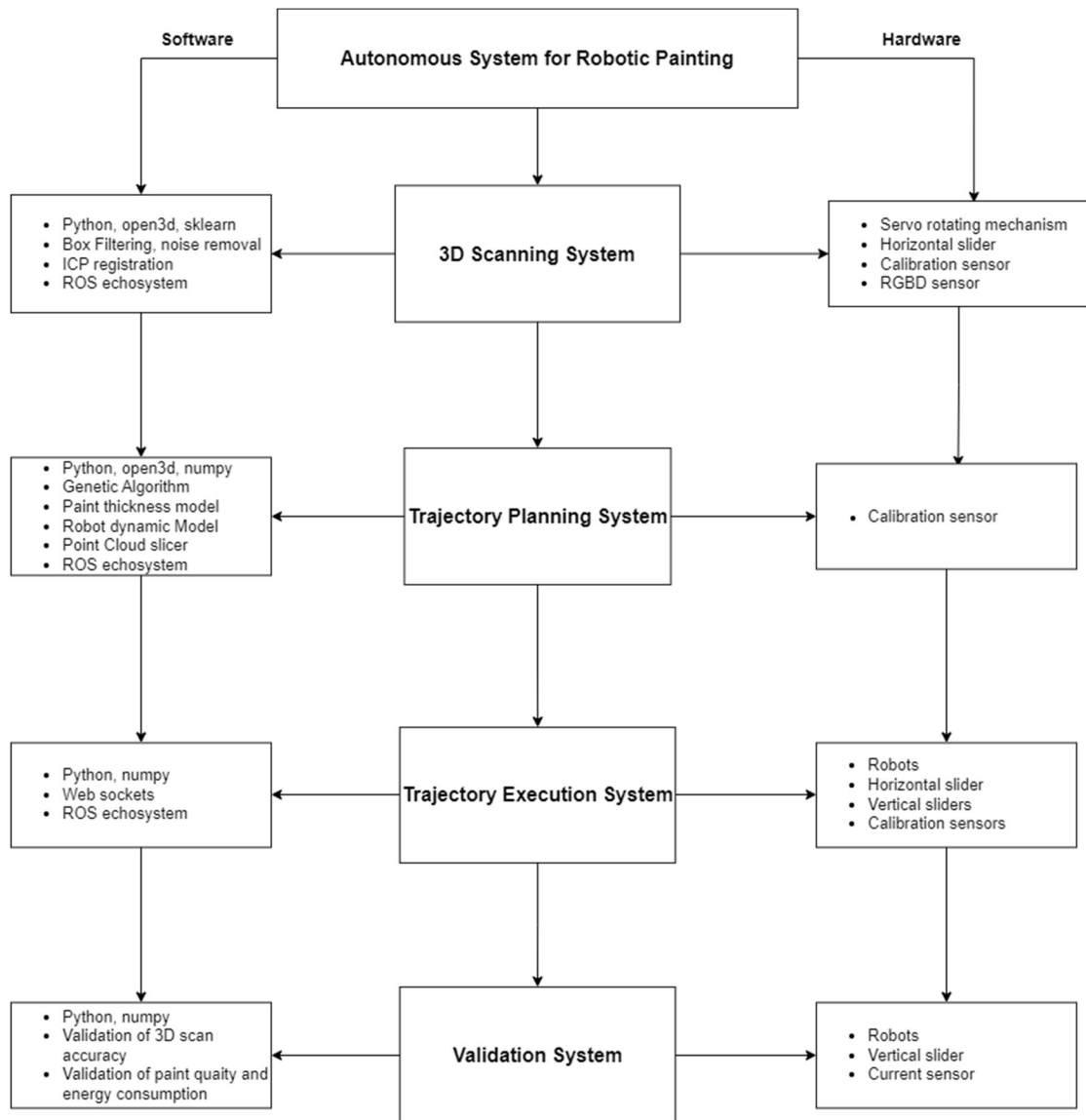
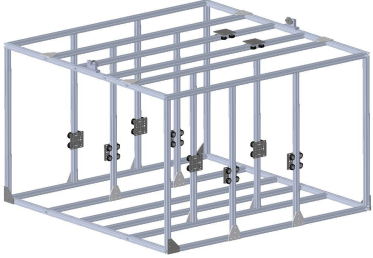
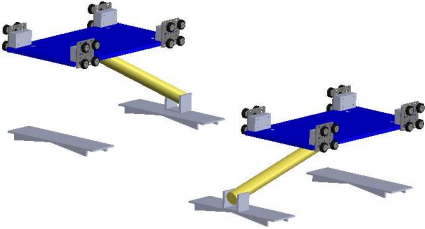
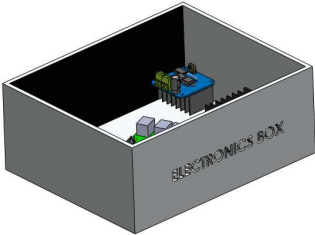
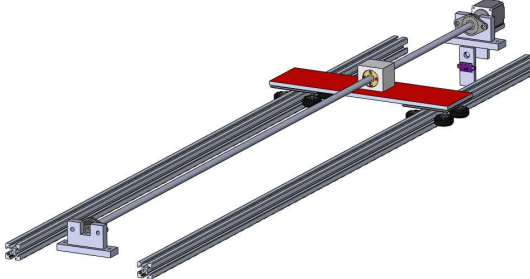
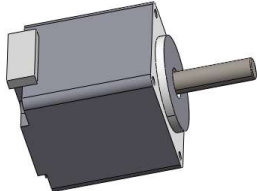
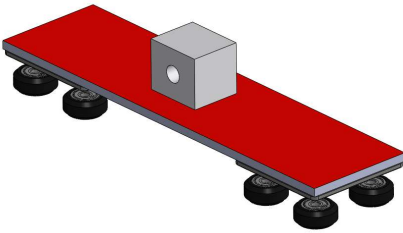
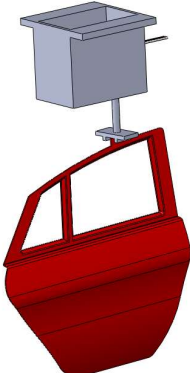
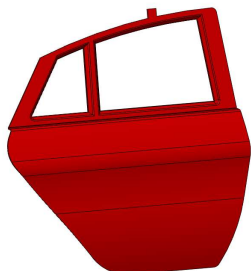



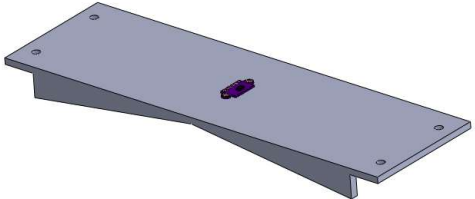
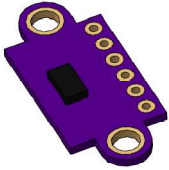
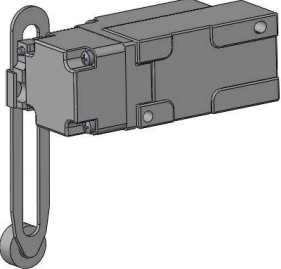
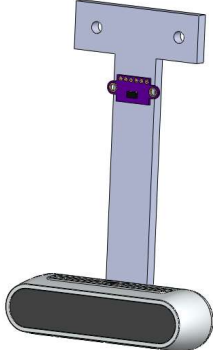
Figure 5.1: Software and hardware development methodology.

## 5.2.1 Hardware development

Table 5.1: Hardware components breakdown with component IDs, descriptions, and the corresponding CAD models.

ID	Component Description	CAD Model
1	Aluminum Framing. Used for building the structure of the entire system. It also contains corner brackets, T-brackets, and gantries for achieving the smooth linear motion.	
2	Vertical Sliding mechanism. It contains 2 linear actuators, a support base for lifting the robots and a base mount plate for securing the linear actuators. For the position feedback, distance sensors are installed.	
3	Electronics box for keeping the electrical components. It contains a raspberry pi controller, an Arduino, 2 motor controllers for the linear actuators, a current sensor, and a stepper driver for controlling the stepper motor.	

4	Horizontal sliding mechanism. It contains a threaded rod, two guide rails with linear gantries, bearings and bearing supports for the threaded rod, a stepper motor for driving the mechanism and a distance feedback sensor.	
5	Stepper motor for moving the horizontal slider [74].	
6	Horizontal slider jockey. It contains a lead screw head connected to the base plate which is then connected to the linear gantries for moving along the guide rails.	
7	Servo rotating mechanism for 3D scanning system. The servo [67] is connected to the object via a gripper that can be tightened and loosened. The object is a car door as illustrated in the CAD.	
8	A downscaled CAD model of a car door for 3D scanning and trajectory planning.	

<p><b>9</b></p>	<p>Two 3DOF robots for controlling the x, y, z location of the end-effector [75]. The robot combined with the vertical sliding mechanism gives a total of 4 DOF for executing the trajectory over the surface of the complex free-form surface (e.g., car door)</p>	
<p><b>10</b></p>	<p>Position feedback platform for the linear actuators. It has a VLX distance sensor mounted on a plate.</p>	
<p><b>11</b></p>	<p>VL53L0X sensor [66]. It is a time of flight (TOF) sensor for measuring distance. It has a measurement range of 3 cm to 2 m and an accuracy of <math>\pm 1mm</math>.</p>	
<p><b>12</b></p>	<p>A limit switch used for disconnecting the power from the linear actuators [76].</p>	
<p><b>13</b></p>	<p>3D scanning hardware [65]. It has an Intel Real Sense D435 sensor and a TOF sensor for calibration of the rotation axis of the object. The sensors are connected to the system via a 3D-printed mounting plate.</p>	

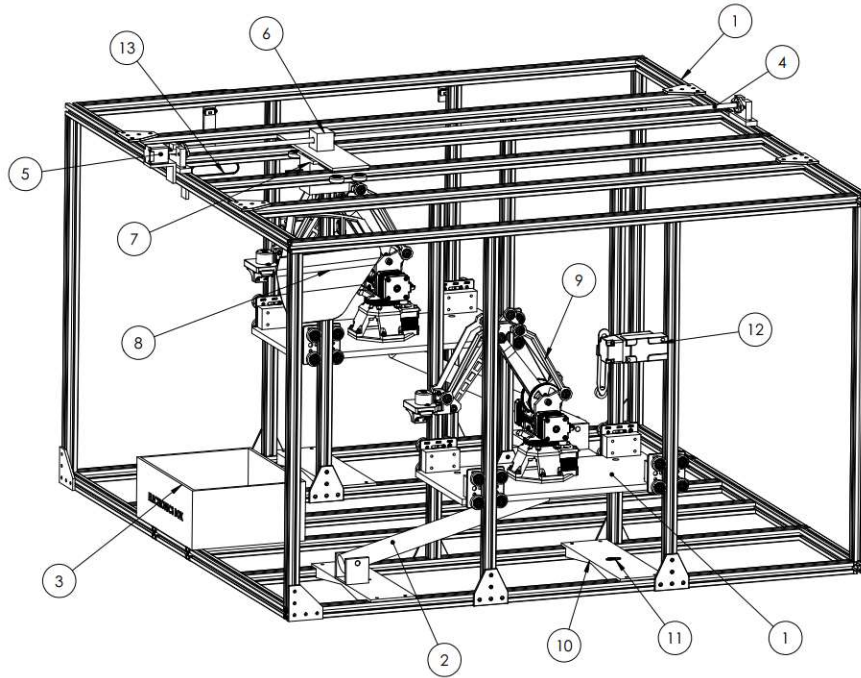


Figure 5.2: CAD schematic of the Integrated System with component IDs.

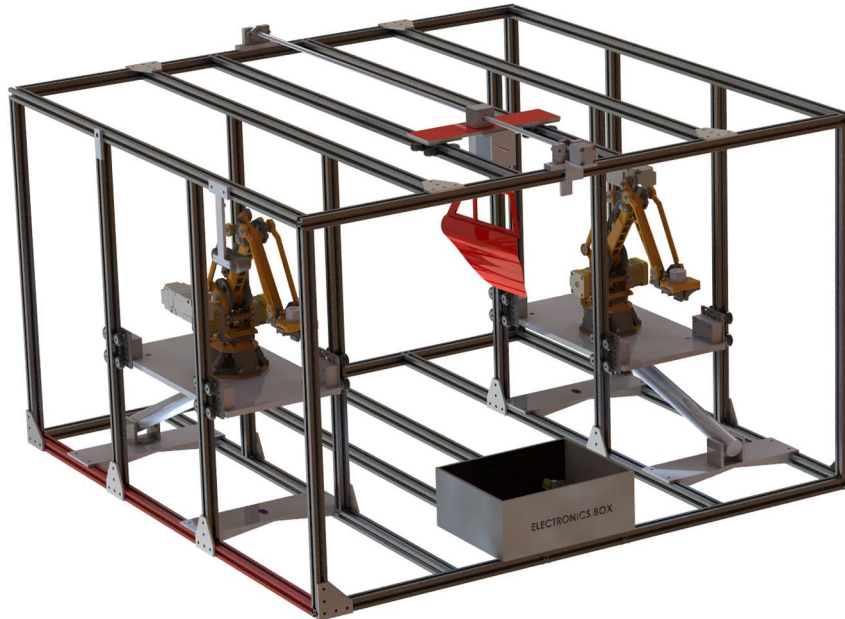


Figure 5.3: 3D rendered CAD model of the Integrated System (isometric view).

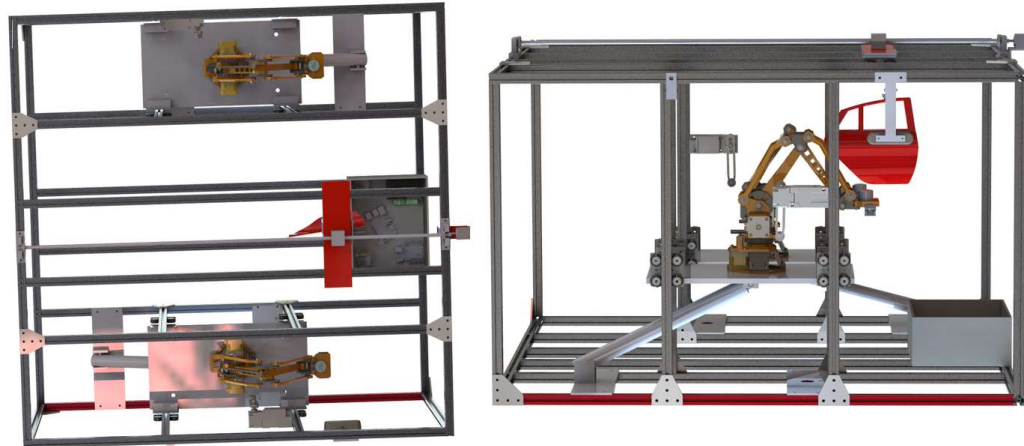


Figure 5.4: 3D rendered CAD model of the Integrated System (top and side view).

### 5.2.2 Software development

After the system is modeled in CAD and fabricated, it is important to devise a software mechanism that achieves the desired purpose of the autonomous system. As such, the software should be able to communicate with the hardware components and optimize the trajectory for the complex free-form surface. Moreover, a GUI (graphical user interface) should also be in consideration to allow the user to interact with the system, make changes to the settings, and load the CAD and trajectory files. The software breakdown and the GUI is shown in Fig. 5.5. The core components of the system are connected via ROS (Robot Operating System) ecosystem [77]. ROS allows for easy communication between software scripts/nodes via topics and services. The ROS MASTER is the main server running all the necessary nodes responsible for trajectory optimization and 3D scanning. Another instance of ROS runs on the Raspberry Pi controller which is directly connected to the hardware. The hardware includes the horizontal sliding and vertical sliding mechanisms, the servo rotation mechanism, the *vlx* sensors for distance monitoring and the D435 sensor for acquiring a depth scan of the object. The raspberry pi is programmed to respond to certain topics via ROS subscriber and publishers. Thus, any sensor can be read by the MASTER node by subscribing to it. Similarly, any change in the actuator state can be published to the raspberry pi node and realized in real time. The web-based GUI is connected to ROS Master and the two robots via web sockets programmed in JS (JavaScript) [78] . The



backbone of the web page is defined by HTML script while the page is styled using CSS and JS acquired from Bootstrap and jQuery [79]. ROS ecosystem is tunneled with the JS using the *roslibjs* script developed by [80]. The two robots receive trajectory commands in the form of x, y, z and location and a time variable for defining the speed. The 3D scanner node is responsible for instantiating a 3D scanning instance on user request from the web GUI. After completion, it stores the scan file into the scan directory locally. Similarly, the trajectory optimizer node when triggered from the GUI, optimizes the trajectory, and stores the results into a NumPy array locally in the *traj* folder.

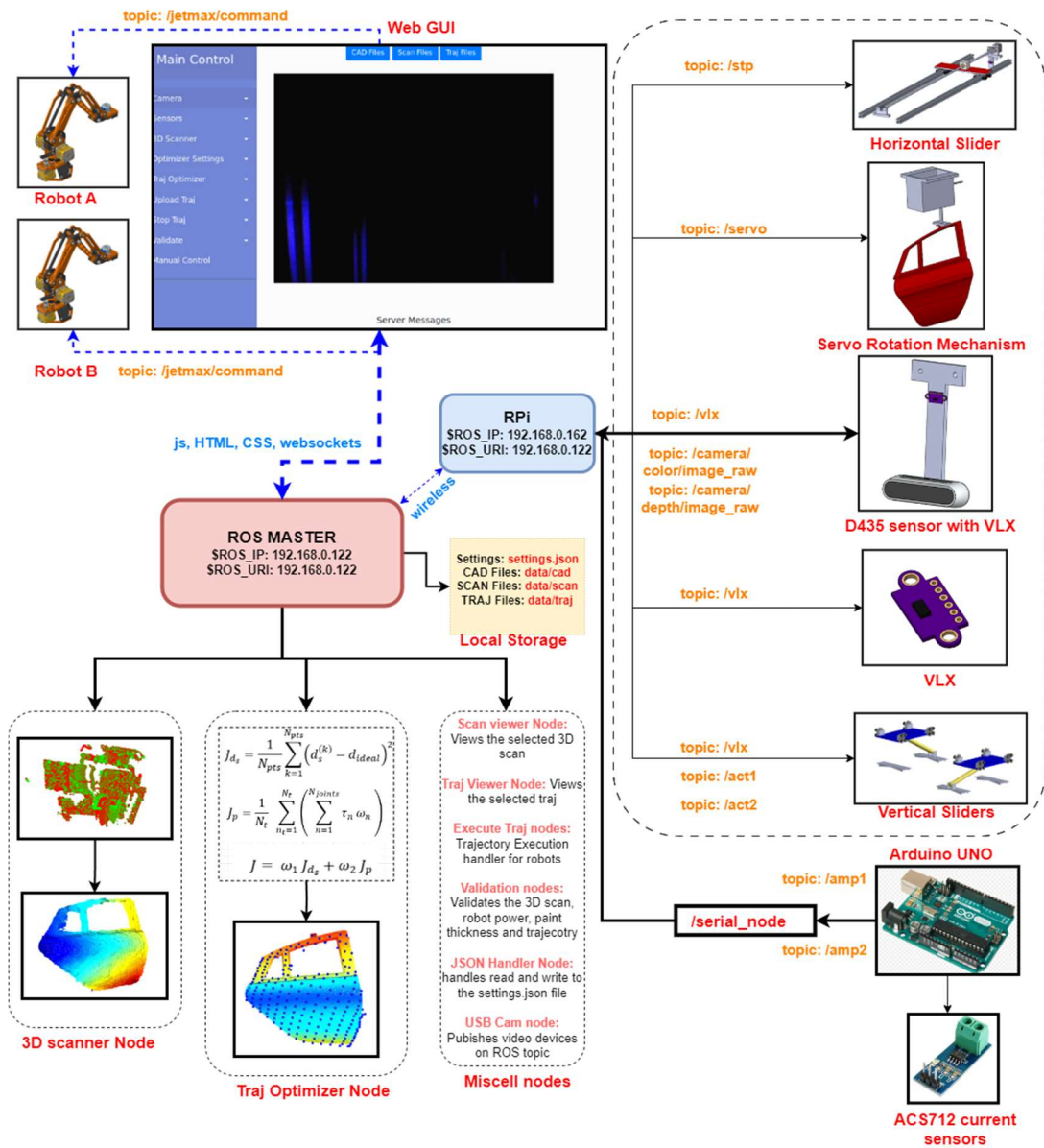


Figure 5.5: Schematic for Software development of the integrated system.

### 5.2.3 Graphical user interface

The GUI is a web-based interface allowing for communication between the user and the software components of the system. The main motivation behind the GUI is the provision of a user-friendly interface to perform the different functions of the system. The web-based GUI has a sidebar navigation menu with interactive buttons for communicating with the system. The main buttons are Camera, Sensors, 3D Scanner, Optimizer Settings, *Traj* Optimizer, Upload *Traj*, Stop *Traj*, Validate and Manual Control. The top navigation bar has 4 buttons including CAD Files, Scan Files, CAD-CAL files and *Traj* Files. These buttons are handled by the JavaScript node and upon clicking will execute the required functionality. The GUI interface is shown in Fig. 5.6.

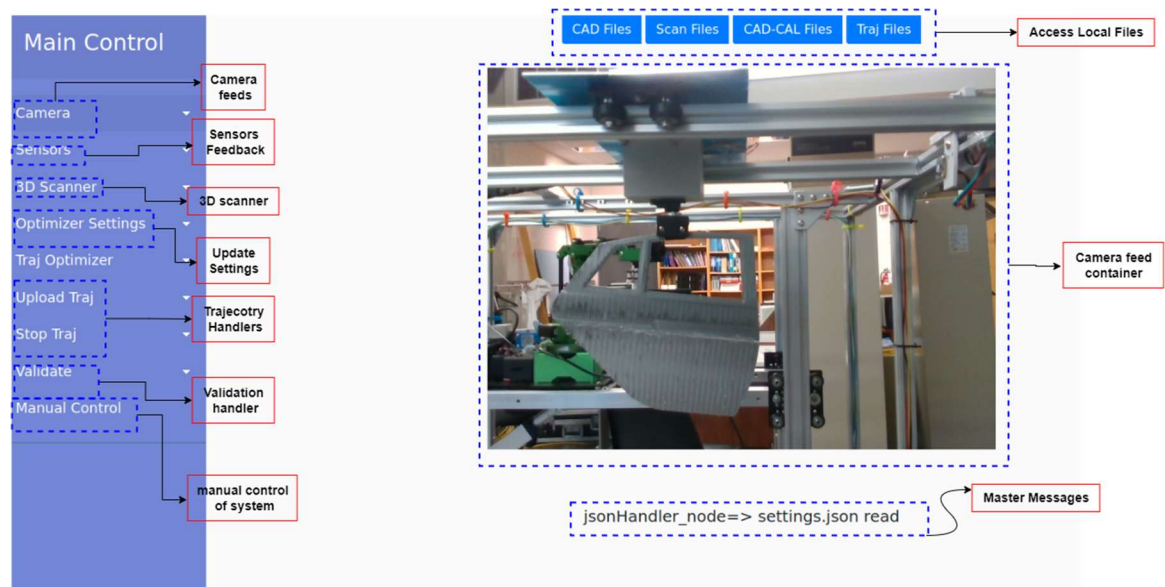


Figure 5.6: Front panel of web-based GUI.

The further breakdown of the GUI interface is shown in Fig. 5.7. The top navigation bar is a clickable dropdown menu showing the available file names on the local storage. Upon clicking either of these three buttons, the JavaScript handler function sends a ping to the JSON handler node at the backend requesting for the File list on the specified directories. The backend node responds with the file list which is updated to view on the drop-down menu. Further, by clicking a filename, it is selected as the current file source for the trajectory planning process. The side bar field, Optimizer Settings, is also a



dropdown menu with form inputs for the paint trajectory planning process. These fields are inserted, and the update button is then clicked to update the settings to the *settings.json* file.

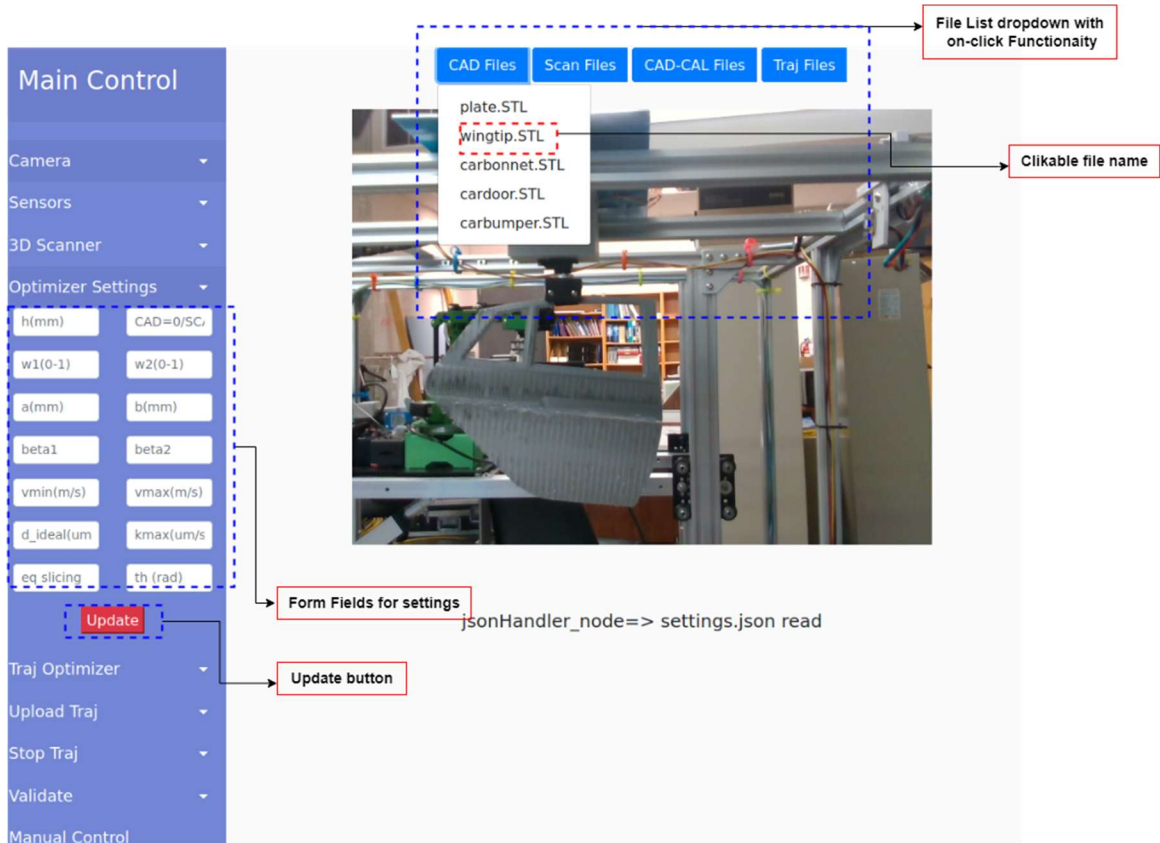


Figure 5.7: Web GUI Optimizer Settings and File System Handler.

The remaining functionalities of the GUI are shown in Fig. 5.8. The *Traj* Optimizer button is further subdivided into two buttons. One is used for optimizing the trajectory while the other is used for viewing the selected trajectory. Before trajectory optimization, the CAD/SCAN field of the Optimizer Settings should be selected 0 or 1 if a CAD or SCAN is to be used respectively. The selected CAD or SCAN file is then considered by the backend for trajectory planning. Similarly, the Upload and Stop *Traj* buttons are used to invoke the trajectory handler node which can start and stop trajectory upload to the robots at any given time. The Validate button has 3 sub fields for evaluating the 3D scan accuracy of the selected CAD and SCAN files, energy, and paint quality for the selected trajectory file. Fig. 5.8 also shows the Camera field dropdown for selecting a given camera feed, the

Sensors dropdown for viewing the sensor readings, and the 3D scanner button for starting the 3D scan and viewing it.

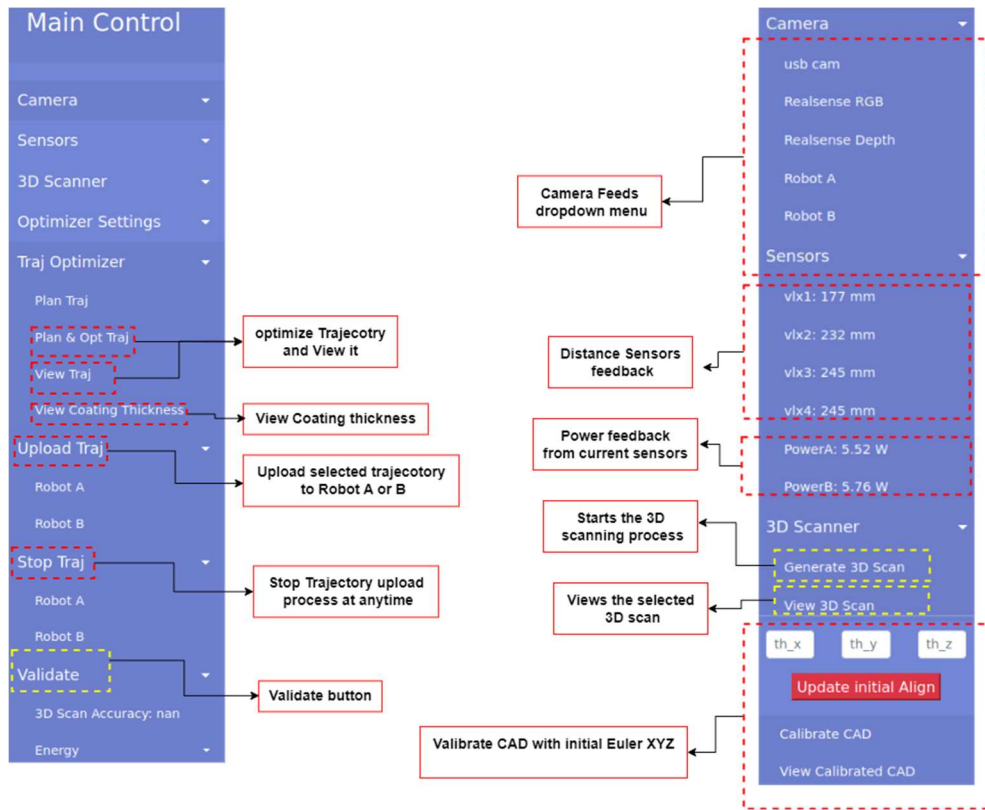


Figure 5.8: Web GUI miscellaneous buttons and functions.

The software packages and custom programs (18 Python scripts) are shown below for reference.

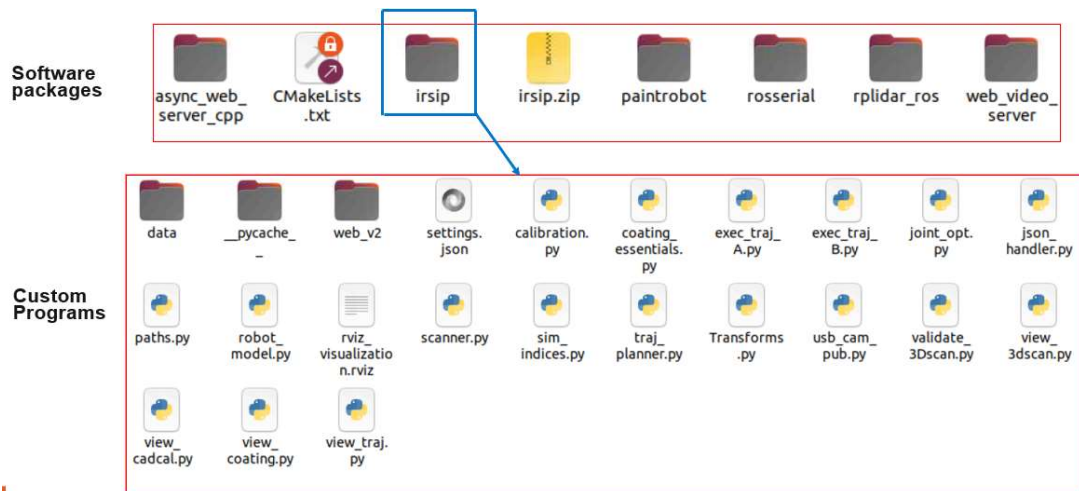


Figure 5.9: Software packages and custom Python scripts.

## 5.2.4 ROS RQT graph

The ROS RQT graph shows the relationships between the topics, services, and nodes running. An RQT graph of the system running with all nodes is shown in Fig. 5.10. The topics are represented by the square boxes and the nodes by an oval shape.

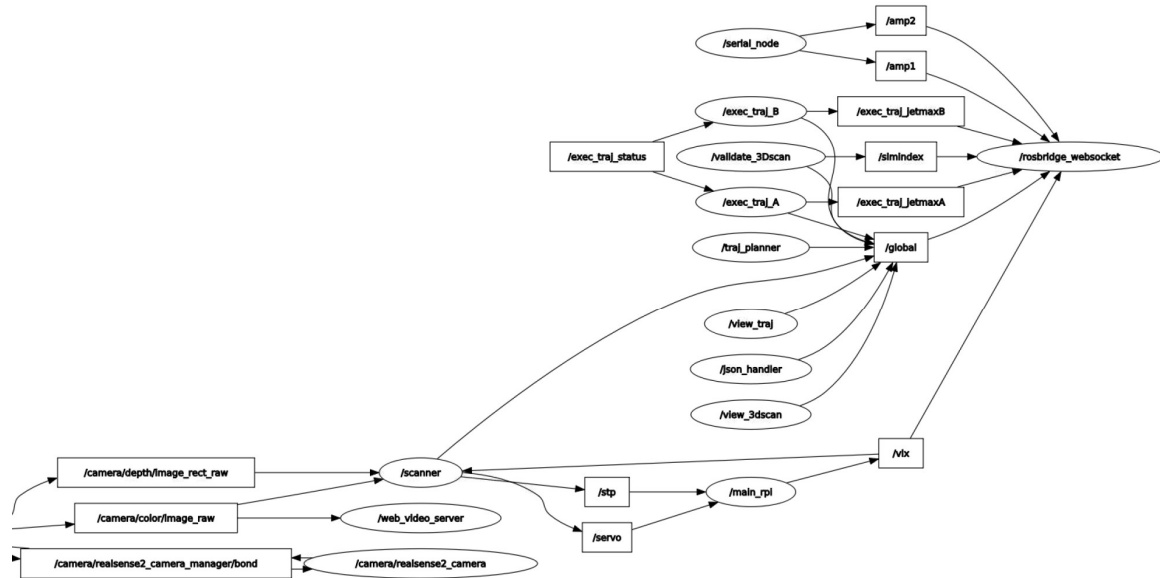


Figure 5.10: ROS RQT graph.

## 5.3 Conclusion

In this chapter, the hardware and the software development were discussed in detail. The major hardware components include two robots for executing the paint trajectory, an intel realSenseD435 sensor for depth mapping, and the LINUX server for running the software components. A Raspberry Pi controller is also used with its GPIO (General purpose input output) pins connected to the sensors, motors, and actuators via electronic drivers. The system is integrated to form an autonomous system for trajectory planning and optimization of the painting process over complex free-form surfaces. The software components include the programs responsible for handling the functionalities of the system. A web-based GUI interface that allows for a user-friendly interface with the system was also discussed in detail.

## Chapter 6. Results and Discussions

The results and discussion section will cover the accuracy of the 3D scanning system used for acquiring the geometry of the object, the calibration results for the CAD models, and the optimization results for the trajectory planning process. The trajectories are analyzed based on the coating distribution error, relative coating error, energy consumption and time. During the energy computation of the manipulator, the friction model is not considered to simplify calculations. Similarly, the end-effector wrench loading vector is also assumed zero. To evaluate the energy consumption experimentally, current sensors are installed at the power inlets of the robots. The spraying process model and the robot dynamic model are implemented in Python including the other functionalities defined in chapter 5. Before discussing the results, we define the parameters for the spraying process model, the robot model, and the genetic optimizer used in the theoretical and experimental analysis.

### 6.1 Spraying process, robot, and optimizer parameters

The spraying process can be modelled using the parameters  $a$ ,  $b$ ,  $\beta_x$ ,  $\beta_y$  and  $k_{max}$ . Since we did not have a spray delivery system, these parameters were chosen based on the size of the object and by analyzing the study done by [37]. The robot model contains the kinematic and dynamic parameters. The links are considered cylindrical to simplify the calculations and the mass moment of inertia are computed for the torque computation. The gravity vector is in the negative z-direction of the robot base frame defined in chapter 4. Similarly, other parameters such as the minimum and maximum paint gun speeds, the spray gun height, and the genetic optimizer settings are also defined in Table 6.1. The GA settings are tuned on hit-and-trial by first starting with the default settings in the *pygad* [81] library in Python. The optimizer speed slows down as the number of generations increases. Thus, the settings must be adjusted based on the computational resources available.

Table 6.1: List of spraying process, robot, and optimizer parameters used in analysis.

Parameter	Description	Value
<b><i>Spraying process parameters</i></b>		
$a$	Ellipse longer side for the coating model	15 mm
$b$	Ellipse shorter side for the coating model	5.6 mm
$\beta_x$	Coating distribution beta along the X direction of ellipse	2.3
$\beta_y$	Coating distribution beta along the Y direction of ellipse	4.5
$k_{max}$	Coating deposition rate	50.0 $\mu\text{m/s}$
$d_{ideal}$	Desired coating thickness	20 $\mu\text{m}$
$v_{min}$	Minimum speed of the spray gun	3 mm/s
$v_{max}$	Maximum speed of the spray gun	15 mm/s
$h$	Spray gun height from the surface	10 mm
<b><i>Robot model parameters</i></b>		
$d_{stroke}$	Link 0 stroke length	254 mm
$L_1$	Manipulator Link 1 length	92.54 mm
$L_2$	Manipulator Link 2 length	128.4 mm
$L_3$	Manipulator Link 3 length	144.8 mm
$M$	Manipulator Link 0 mass	2.5 kg
$m_1$	Manipulator Link 1 mass	0.5 kg
$m_2$	Manipulator Link 2 mass	0.5 kg
$m_3$	Manipulator Link 3 mass	0.5 kg
<b><i>Optimizer Parameters</i></b>		
$\omega_1$	Scaling factor for mean squared error	0.40
$\omega_2$	Scaling factor for coating deviation error	0.20
$\omega_3$	Scaling factor for mean energy consumption	0.20
$\omega_4$	Scaling factor for mean trajectory time	0.20
$\epsilon$	Hyper-parameter in the fitness function	1.0
$r_m$	Mutation rate in GA	0.1
$c_{type}$	Crossover type in GA	Two points
$m_{type}$	Mutation type in GA	Random
$N_{parents}$	Number of mating parents in GA	2
$N_{gen}$	Number of generations in GA	25
$N_{sol}$	Number of solutions per population in GA	2



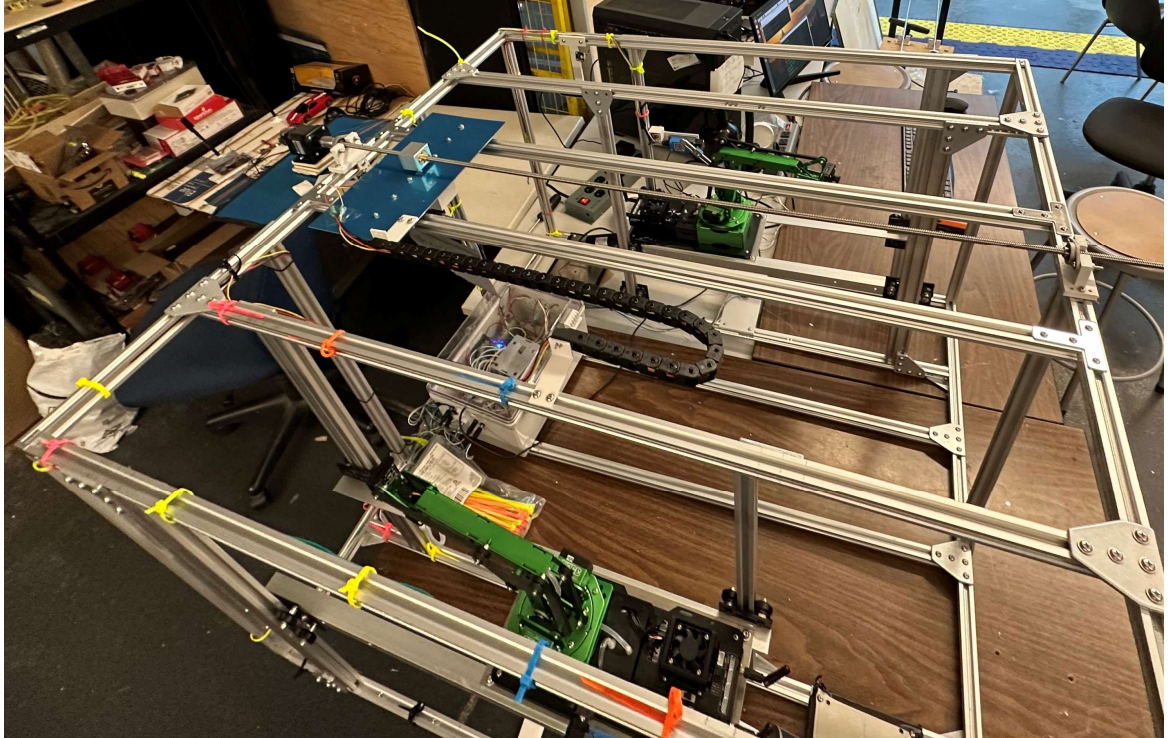


Figure 6.1: Experimental setup in the laboratory.

## 6.2 3D scanning and CAD calibration results

The 3D scanning system is used to generate surface point clouds of the object under investigation. After the point clouds are obtained for each rotational index ( $30^\circ$ ), they are filtered and raw aligned using the transformations defined earlier in Chapter 3. ICP is then used to fine-align and transform the point clouds into one common reference frame. Additionally, statistical noise removal is applied if the point cloud has any residual noise as defined in Chapter 3. Three objects are scanned, and their corresponding CAD models are calibrated (transformed) into the camera reference frame  $\{C\}$  for trajectory planning. Fig. 6.2 shows selecting a scan file from the file list button in GUI and viewing it by clicking the view 3D scan button. Similarly, Fig. 6.3 shows performing the calibration using the GUI interface of the system. Table 6.2 shows the summary of the scanned models and their corresponding calibrated CAD in the camera frame  $\{C\}$ .

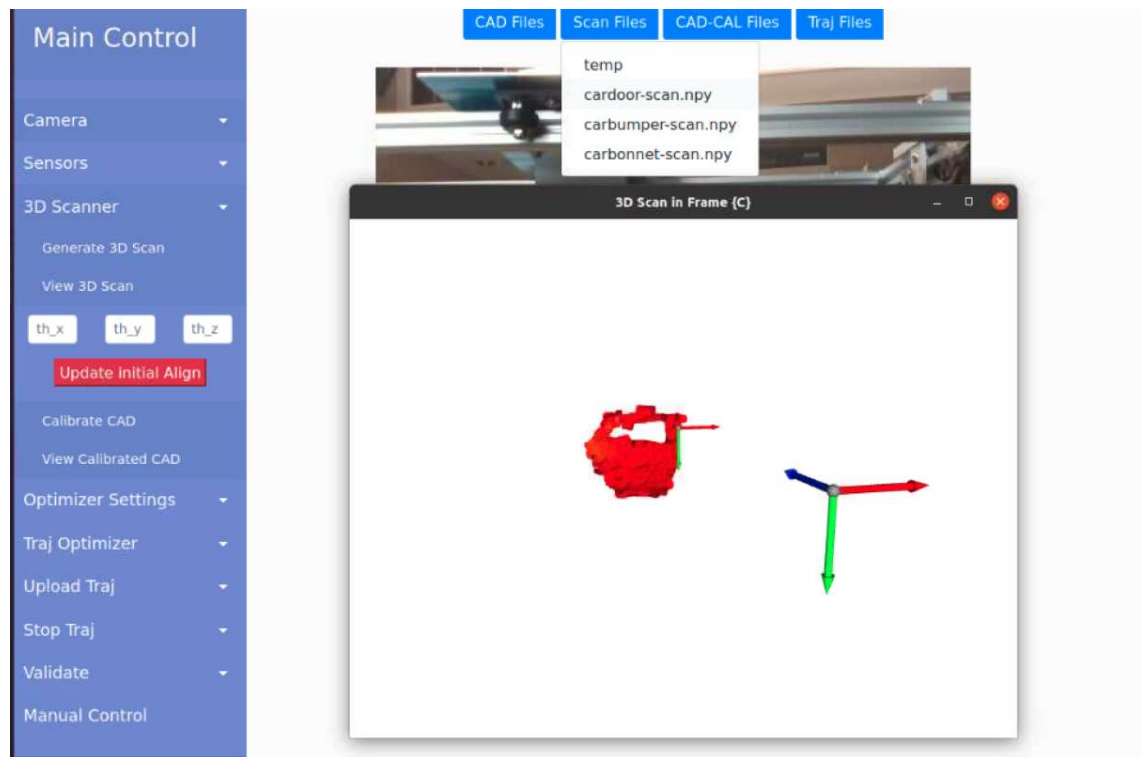


Figure 6.2: Selecting 3D scan and viewing it in the GUI.



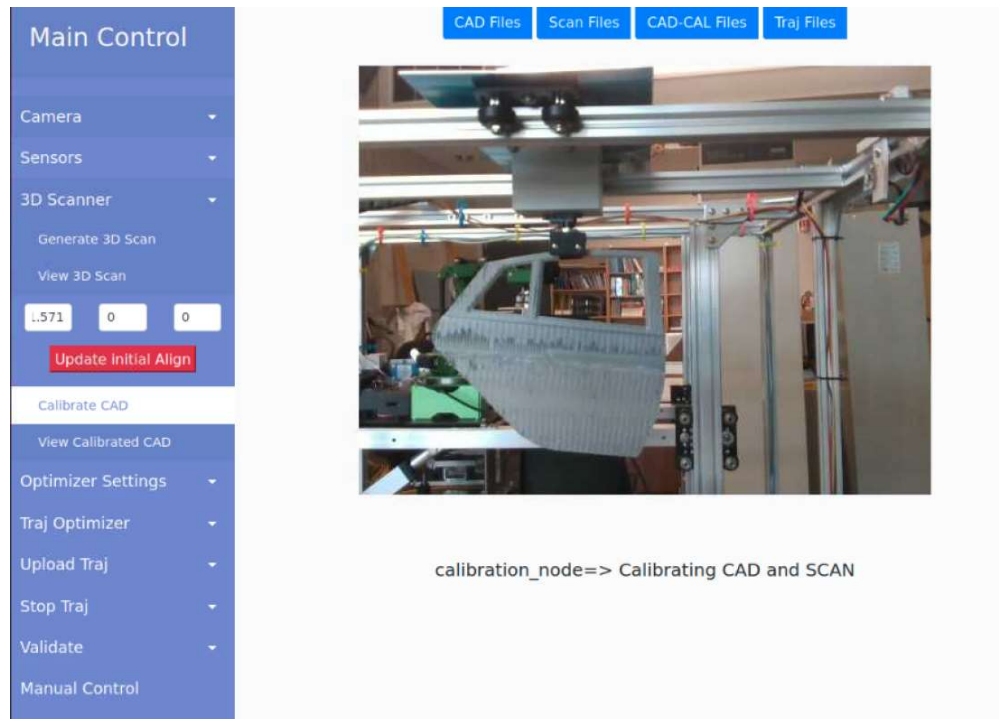
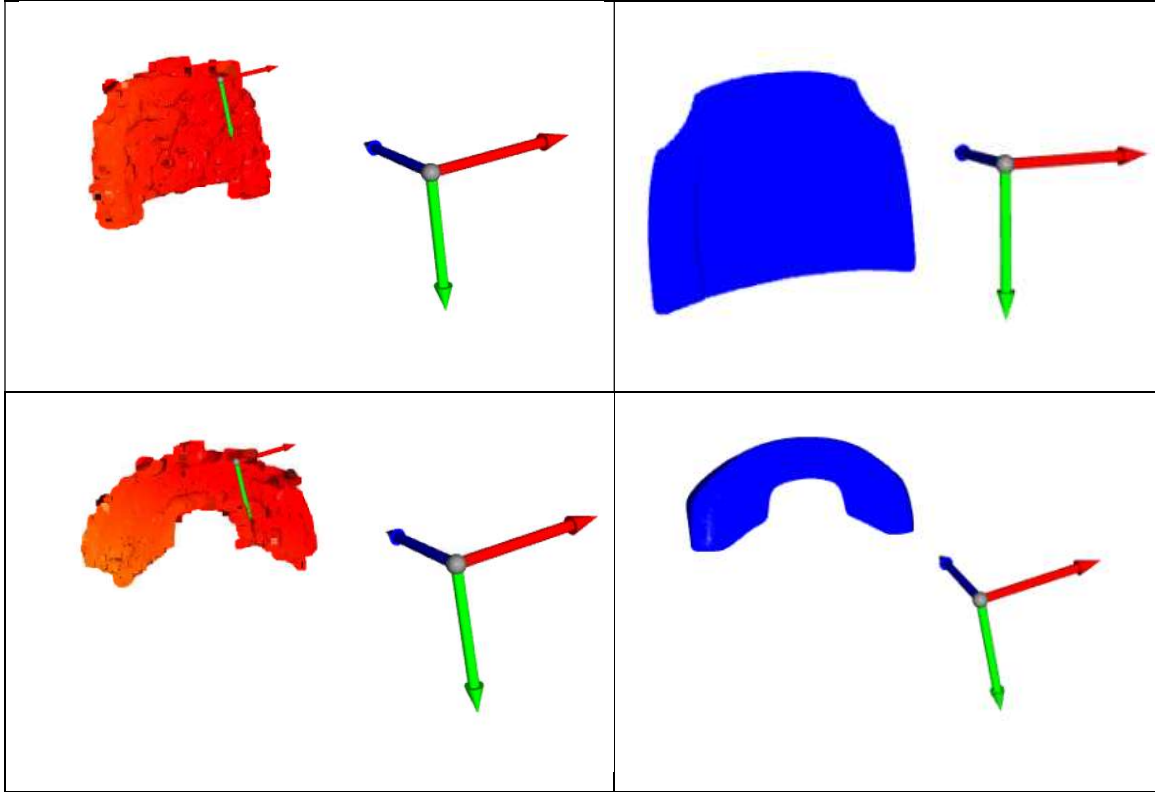


Figure 6.3: Calibrating the 3D scan and the corresponding CAD file in the GUI.

Table 6.2: Scanned models and their corresponding CAD calibrated in frame  $\{C\}$ .

Scanned model in frame $\{C\}$	Calibrated CAD in frame $\{C\}$





The accuracy of the scanning process is evaluated using the  $D_1$  and  $D_2$  metrics defined earlier in chapter 3. The  $D_3$  and  $A_3$  metrics are omitted since their computation depends on angles in the point cloud and generates errors in the inverse cosine frequently. The  $D_1$  and  $D_2$  metrics are computed between the scanned models and their corresponding CAD models. It is revealed that the car door is captured with 95% accuracy, the car hood with 93%, and the car bumper with 92% accuracy. The geometric signatures are summarized in Table 6.3 while the density plots are shown in Fig. 6.4 to Fig. 6.6.

Table 6.3: Similarity scores between the 3D scanned models and the corresponding CAD.

	<b>D<sub>1</sub> score</b>	<b>D<sub>2</sub> score</b>	<b>Avg score</b>
<b>Car door</b>	0.9639	0.9434	0.9536
<b>Car hood</b>	0.9524	0.9228	0.9376
<b>Car bumper</b>	0.9488	0.9082	0.9285

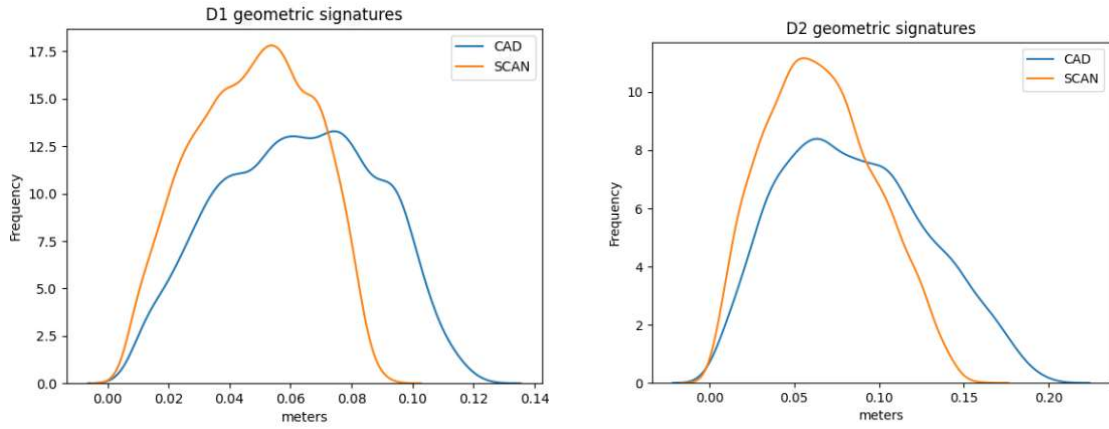


Figure 6.4:  $D_1$  and  $D_2$  density distributions for car door.

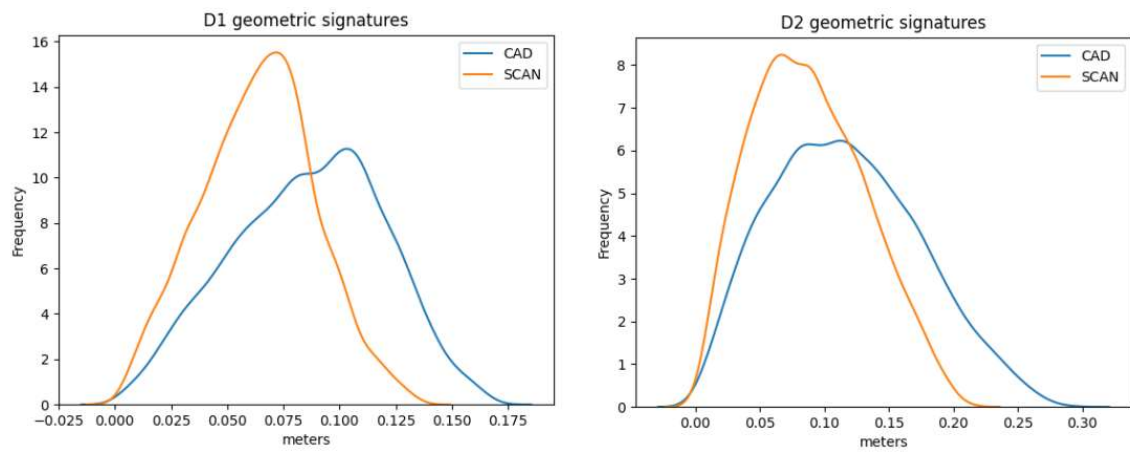


Figure 6.5:  $D_1$  and  $D_2$  density distributions for car hood.

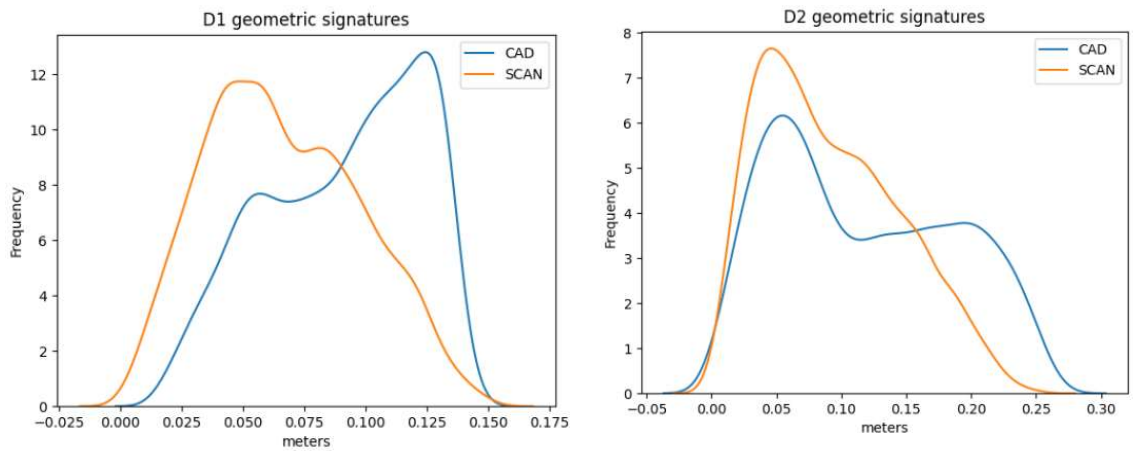


Figure 6.6:  $D_1$  and  $D_2$  density distributions for car bumper.

### 6.3 Optimal paint trajectory planning for a car door

A downscaled version of the car door is considered for trajectory planning and optimization. The CAD model is calibrated with the scanned model to ensure the accurate position and orientation of the surface in the camera and robot frame as defined in Chapter 3. The calibrated CAD model is loaded into Python [61] and converted to eigen coordinate system by applying PCA. The GA is then run for each slice along the slicing direction of the CAD point cloud until no slices are left. The optimizer runs for equidistant slicing ( $\delta = a$ ) and non-equidistant slicing ( $\delta \in [a \ 2a]$ ) and the results are stored for each slice. For ease of analysis, the slicing direction  $\theta$  is discretized into 4 values including  $0^\circ$ ,  $30^\circ$ ,  $60^\circ$  and  $90^\circ$ . The results include analyzing the mean coating thickness, energy per slice, coating distribution error, relative coating error, GA fitness, slice widths, slice speeds, and inverse kinematic configurations plotted against slice numbers. Additionally, the planned trajectory in the eigenframe and the coating thickness at each patch are visualized using color intensities proportional to the value of coating thickness. The coating thickness is mapped to color intensities such that the brighter green color represents high coating thickness and vice versa. The axes of the frame  $\{EF\}$  are shown by red, green, and blue for x, y, and z respectively as shown in Fig. 6.7.

#### 6.3.1 Results for slicing direction $\theta = 0^\circ$

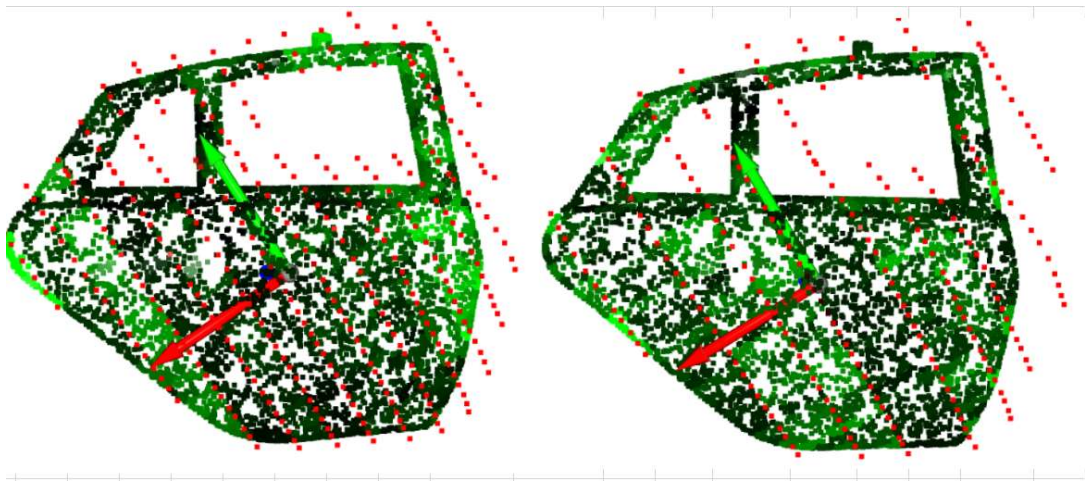


Figure 6.7: Coating thickness and planned trajectory of a car door in  $\{EF\}$  for slicing direction  $\theta = 0^\circ$  (left: equidistant slicing, right: non-equidistant slicing).



Figure 6.8: Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car door:  $\theta = 0^\circ$ ).

### 6.3.2 Results for slicing direction $\theta = 30^\circ$

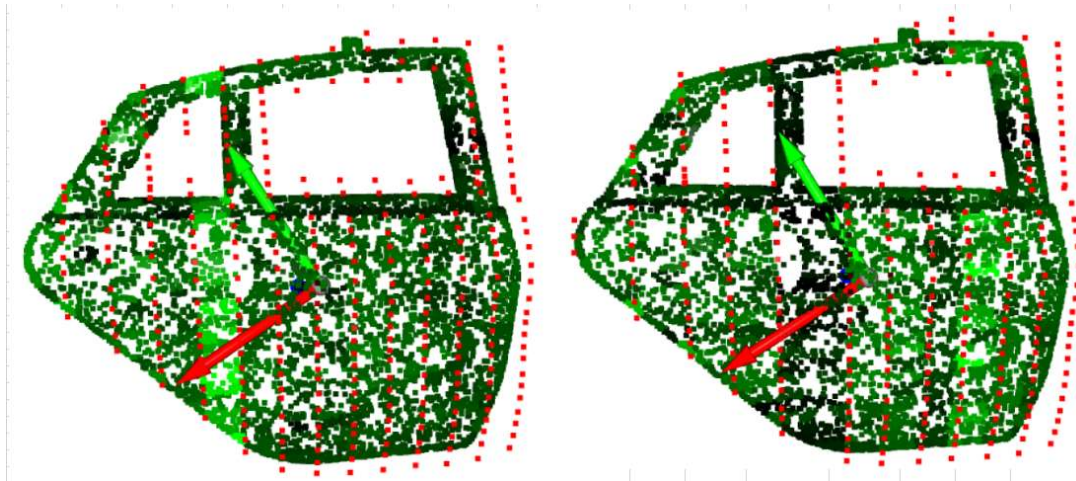
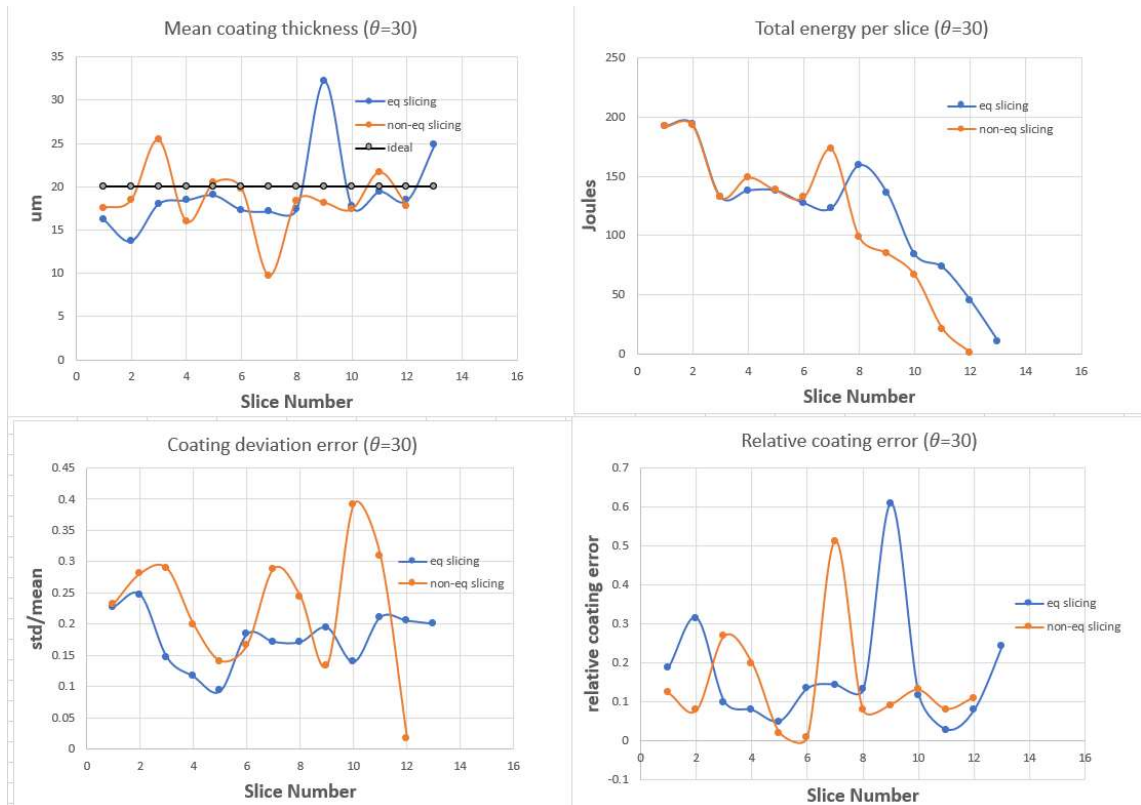


Figure 6.9: Coating thickness and planned trajectory of a car door in {EF} for slicing direction  $\theta = 30^\circ$  (left: equidistant slicing, right: non-equidistant slicing).





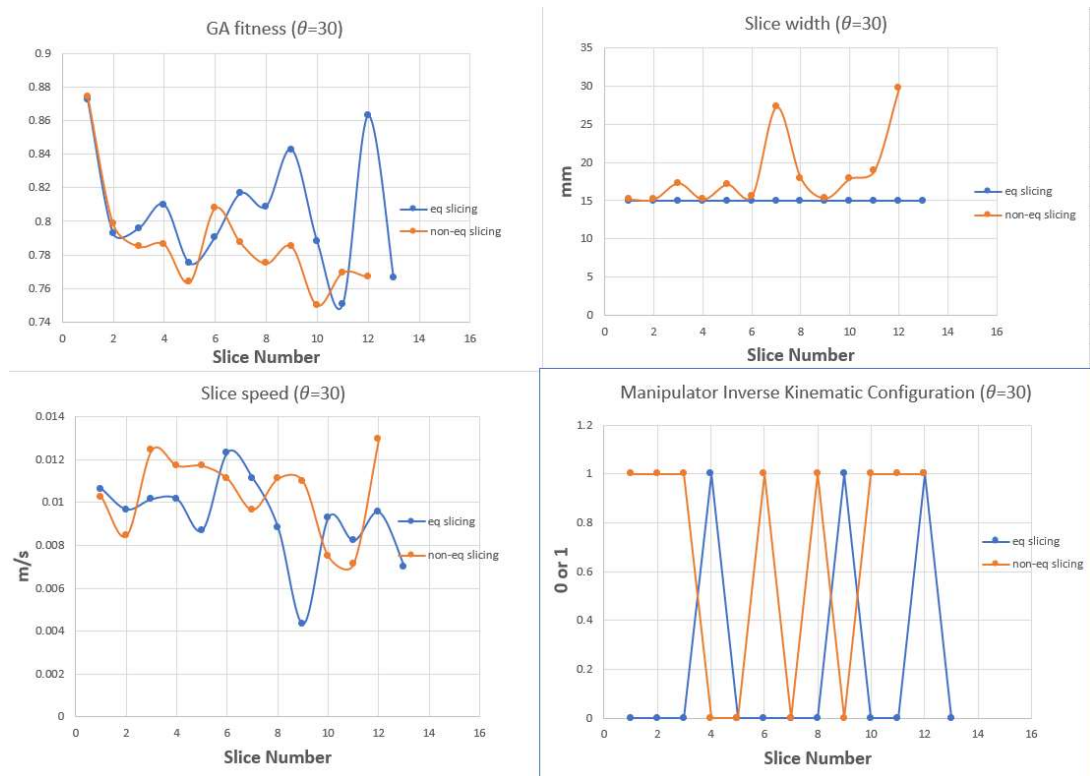


Figure 6.10: Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car door:  $\theta = 30^\circ$ ).

### 6.3.3 Results for slicing direction $\theta = 60^\circ$

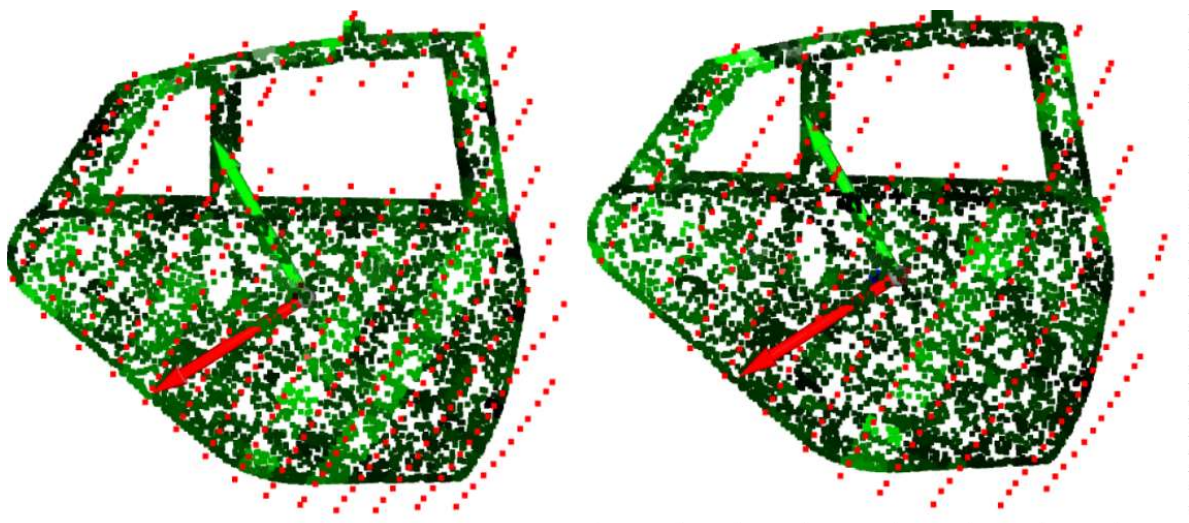


Figure 6.11: Coating thickness and planned trajectory of a car door in  $\{EF\}$  for slicing direction  $\theta = 60^\circ$  (left: equidistant slicing, right: non-equidistant slicing).

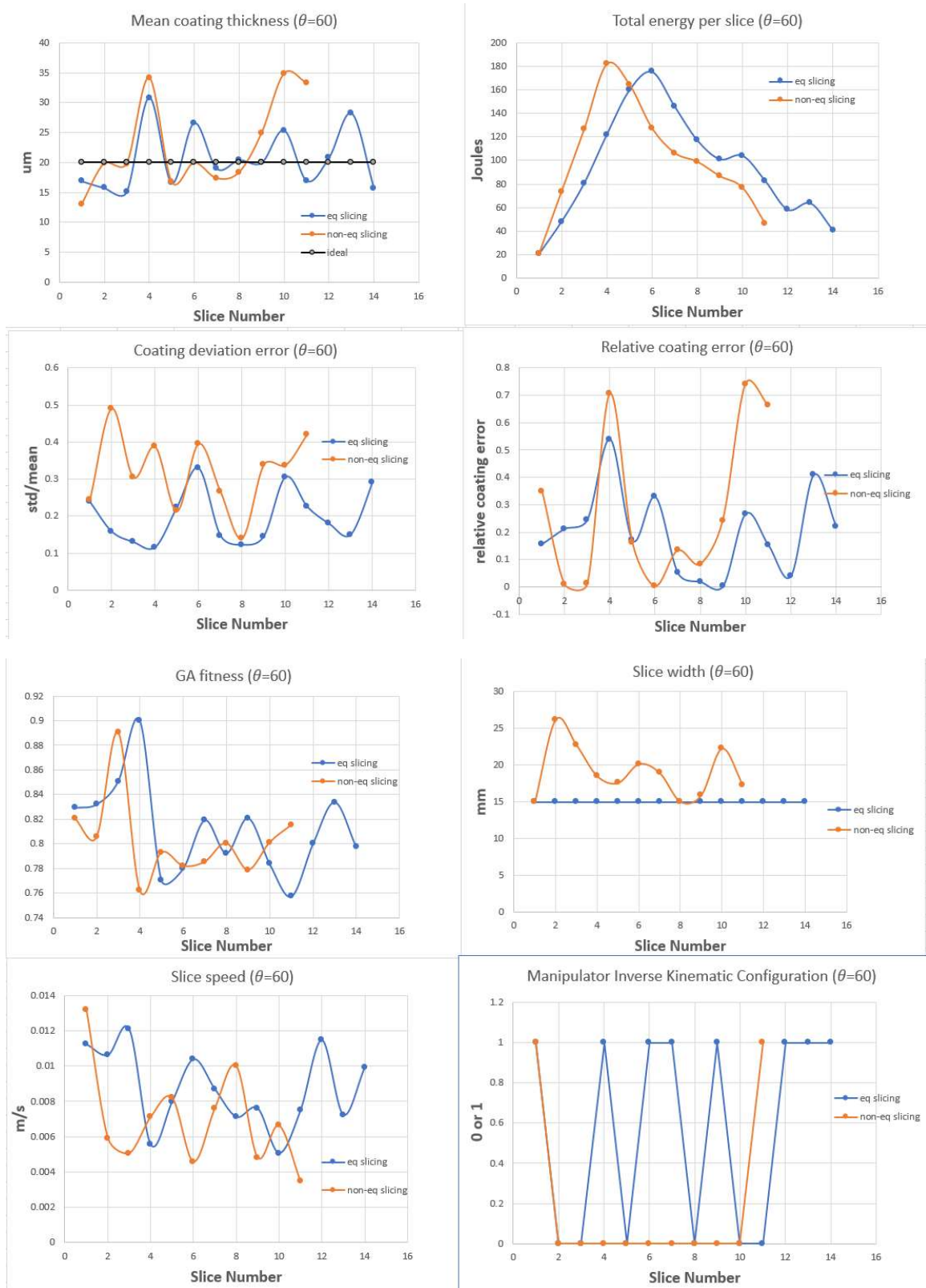


Figure 6.12: Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car door:  $\theta = 60^\circ$ ).

### 6.3.4 Results for slicing direction $\theta = 90^\circ$

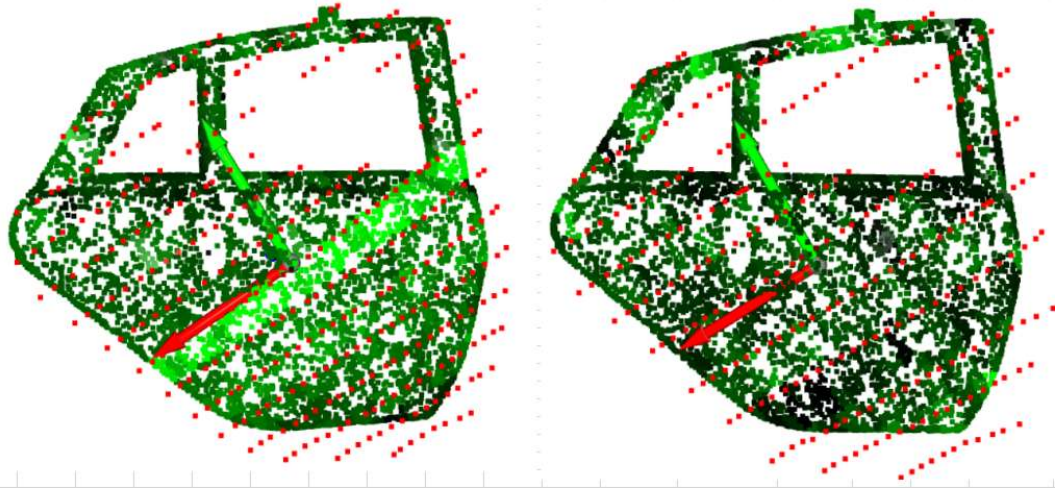
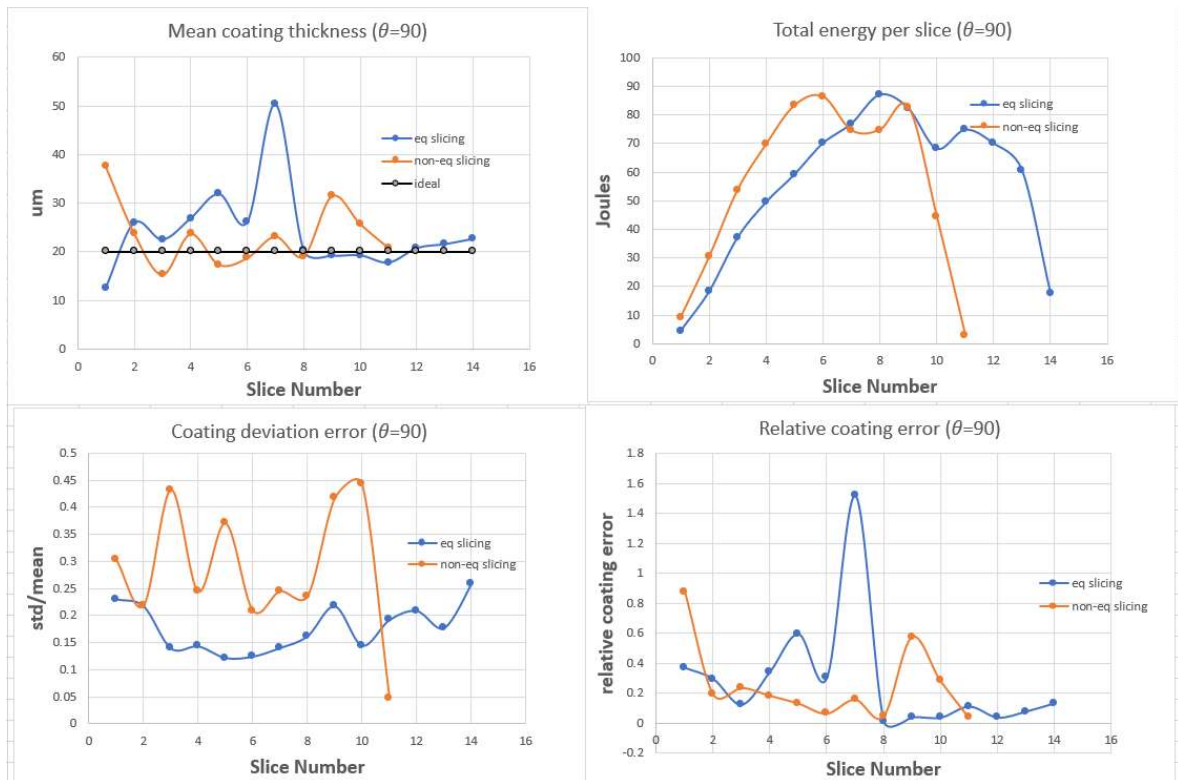


Figure 6.13: Coating thickness and planned trajectory of a car door in  $\{EF\}$  for slicing direction  $\theta = 90^\circ$  (left: equidistant slicing, right: non-equidistant slicing).





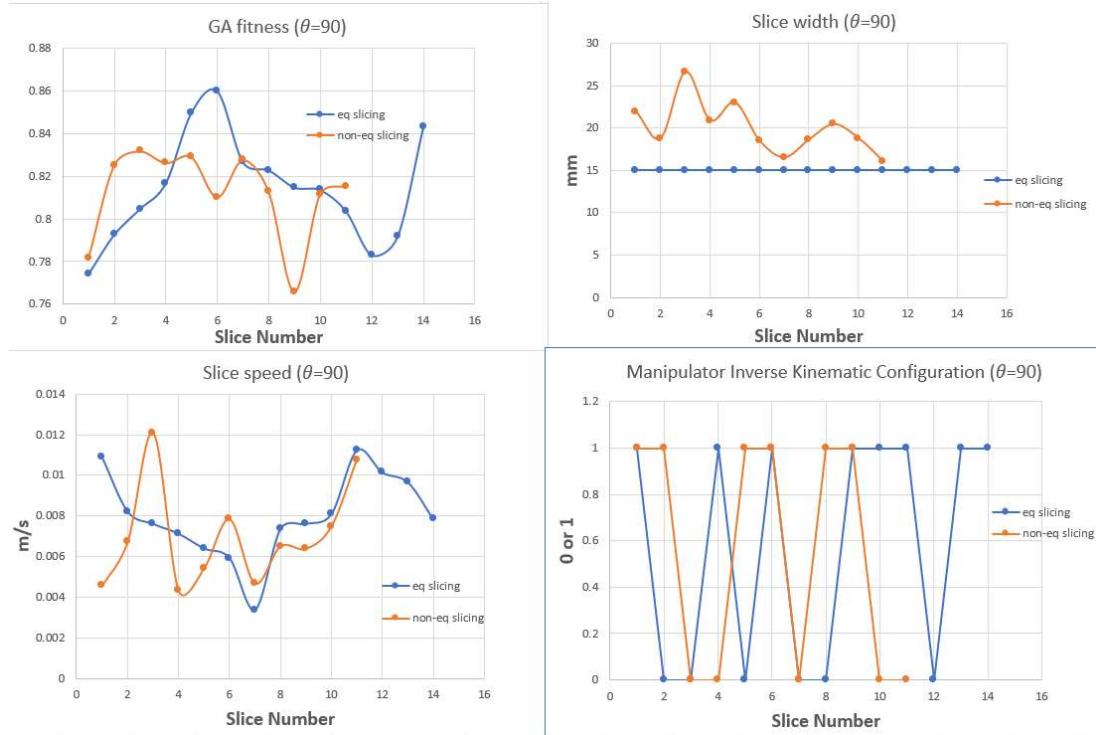


Figure 6.14: Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car door:  $\theta = 90^\circ$ ).

### 6.3.5 Results discussions

The results indicate that a desired coating thickness on a surface can be achieved by specifying the correct spraying parameters and the robot model. The value of the coating thickness, however, is subject to the local geometry of the object, the speed of the paint gun, the slice width and the slicing direction. Thus, variations in the mean coating thicknesses can be seen for both equidistant and non-equidistant slicing. The trajectories are evaluated based on four criteria including energy, time, coating distribution, and relative coating error. These quantities are plotted against the slicing direction as shown in Fig. 6.15. It is observed that the non-equidistant slicing scheme always leads to lower energy consumption since the surface can be covered with fewer slices. A similar trend is also observed for the trajectory time. As far as the coating distribution is concerned, the equidistant slicing scheme gives a more uniform distribution as indicated by the lower deviation errors. The relative coating error is generally lower for the non-equidistant slicing except at a slicing direction of  $60^\circ$ , where it is slightly higher. The most energy efficient

trajectory is obtained at a slicing direction of  $90^\circ$  and by employing a non-equidistant slicing scheme. It consumes a total of 611 J of energy which is 60% lower than the least energy-efficient trajectory. The optimal trajectory for the process time is at a slicing direction of  $30^\circ$  while using a non-equidistant slicing scheme. This trajectory takes 188 s which is 33% lower than the least time-optimal trajectory. The coating distribution is most uniform when employing an equidistant slicing scheme at an angle of  $30^\circ$ . It gives an average coating distribution error of 18% with a mean coating thickness of  $19.21 \mu m$  which is considerably close to the desired thickness of  $20.0 \mu m$ . Finally, the lowest relative coating error (14%) is observed for a non-equidistant slicing scheme at an angle of  $30^\circ$ . This leads to a mean coating thickness of  $18.38 \mu m$ . The summary of results is given in Table 6.4.

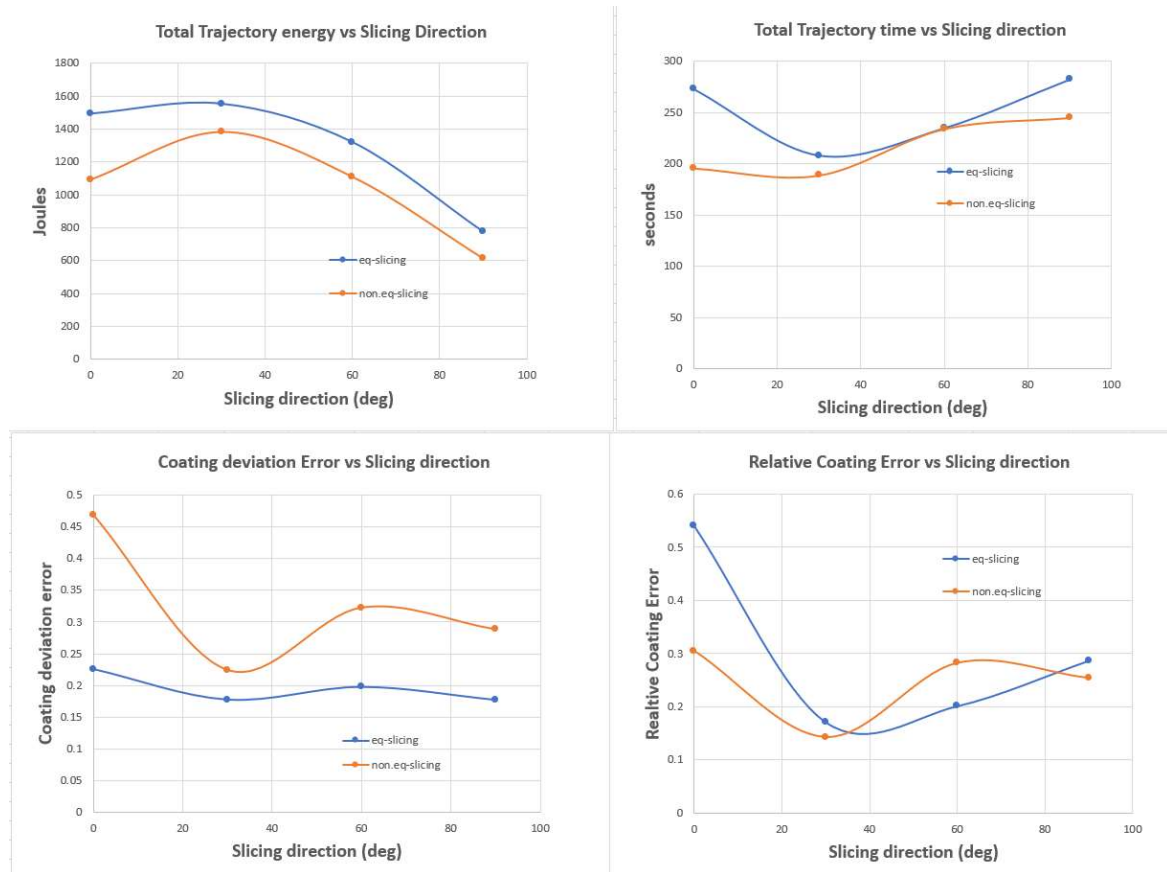


Figure 6.15: Total energy, trajectory time, coating deviation and relative coating error vs slicing direction (car door).

Table 6.4: Results summary for trajectory planning and optimization of a car door for both equidistant and non-equidistant slicing.

	$\theta$	$d_{mean}(\mu m)$	$E_{sum} (J)$	$T_{sum} (s)$	$J_{d_{error}}$	$J_{d_{rel}}$
<b>Eq-slicing</b>	$0^\circ$	28.58	1492.94	273.10	0.23	0.54
	$30^\circ$	19.21	1553.09	207.76	0.18	0.17
	$60^\circ$	20.56	1320.14	234.55	0.20	0.20
	$90^\circ$	24.12	775.75	282.17	0.18	0.29
<b>Non-eq slicing</b>	$0^\circ$	21.91	1092.36	195.47	0.47	0.30
	$30^\circ$	18.38	1382.48	188.54	0.22	0.14
	$60^\circ$	22.91	1109.15	233.77	0.32	0.28
	$90^\circ$	23.34	611.33	244.65	0.29	0.25

## 6.4 Optimal paint trajectory planning for a car hood

### 6.4.1 Results for slicing direction $\theta = 0^\circ$

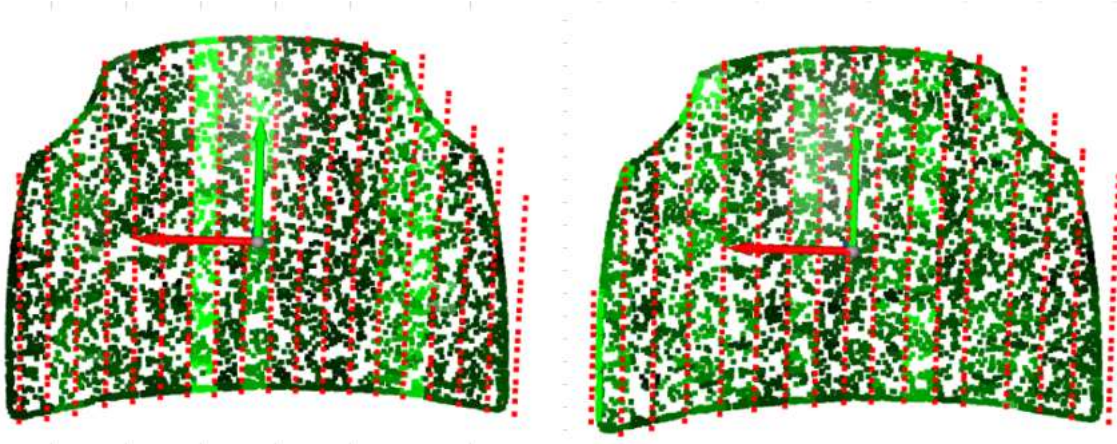
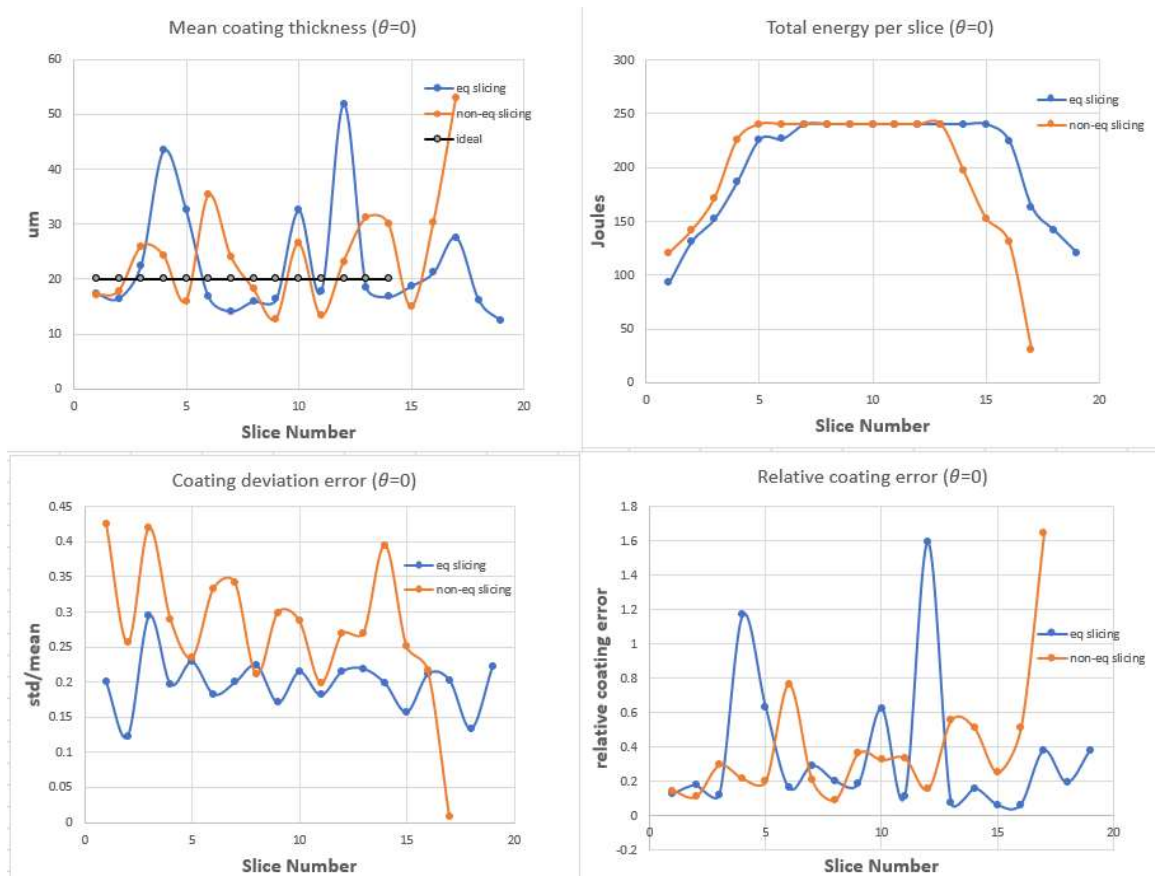


Figure 6.16: Coating thickness and planned trajectory of a car hood in {EF} for slicing direction  $\theta = 0^\circ$  (left: equidistant slicing, right: non-equidistant slicing).



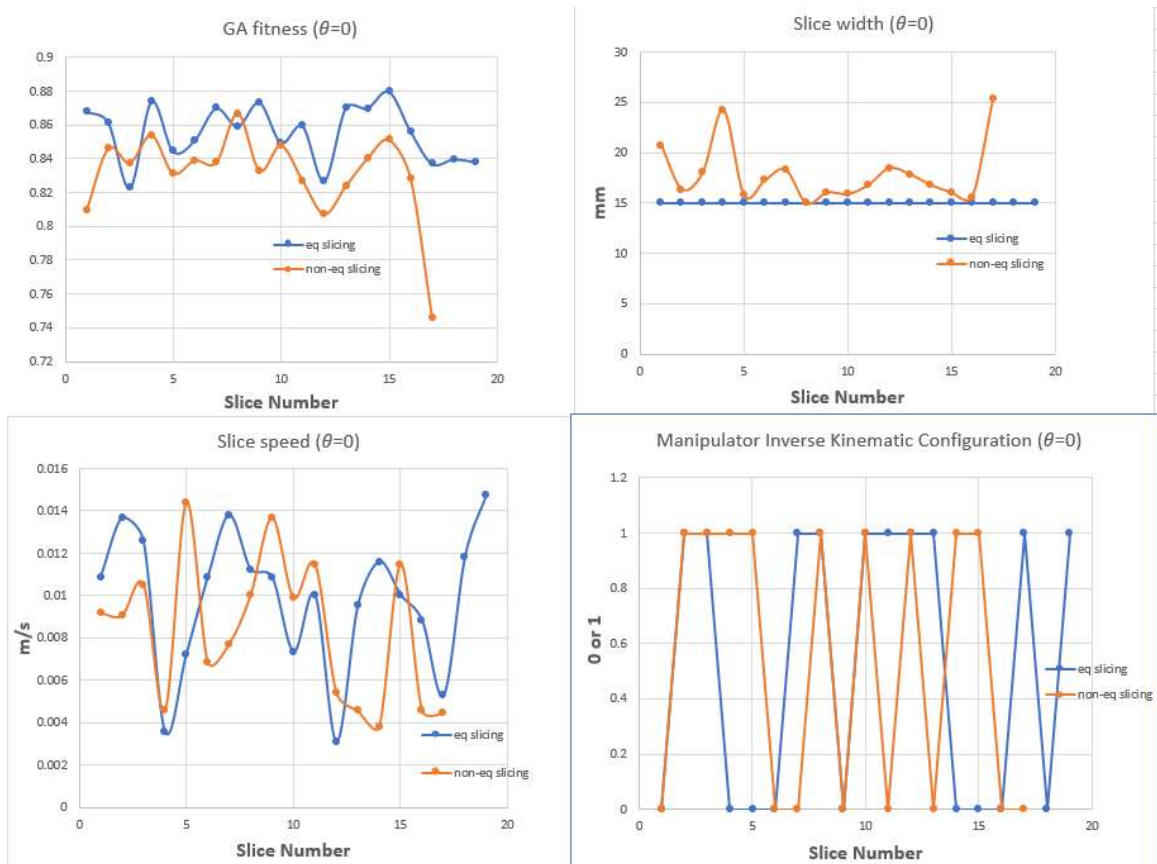


Figure 6.17: Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car hood:  $\theta = 0^\circ$ ).

#### 6.4.2 Results for slicing direction $\theta = 30^\circ$

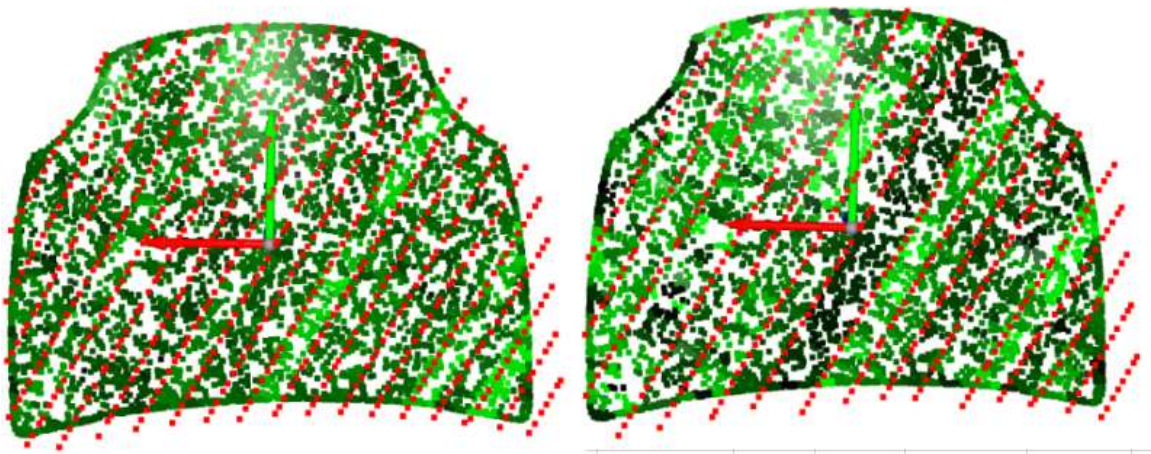


Figure 6.18: Coating thickness and planned trajectory of a car hood in {EF} for slicing direction  $\theta = 30^\circ$  (left: equidistant slicing, right: non-equidistant slicing).



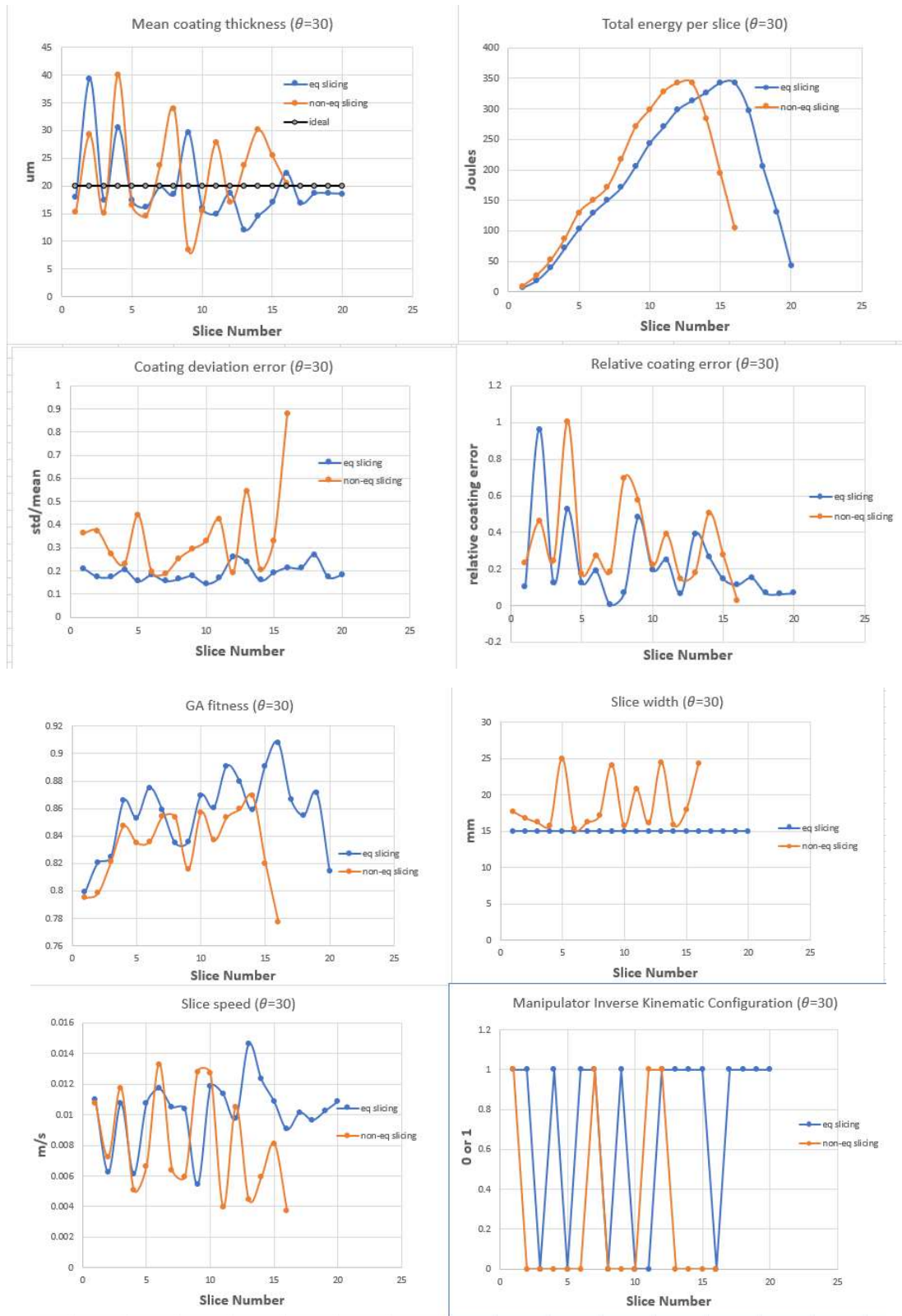


Figure 6.19: Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car hood:  $\theta = 30^\circ$ ).

### 6.4.3 Results for slicing direction $\theta = 60^\circ$

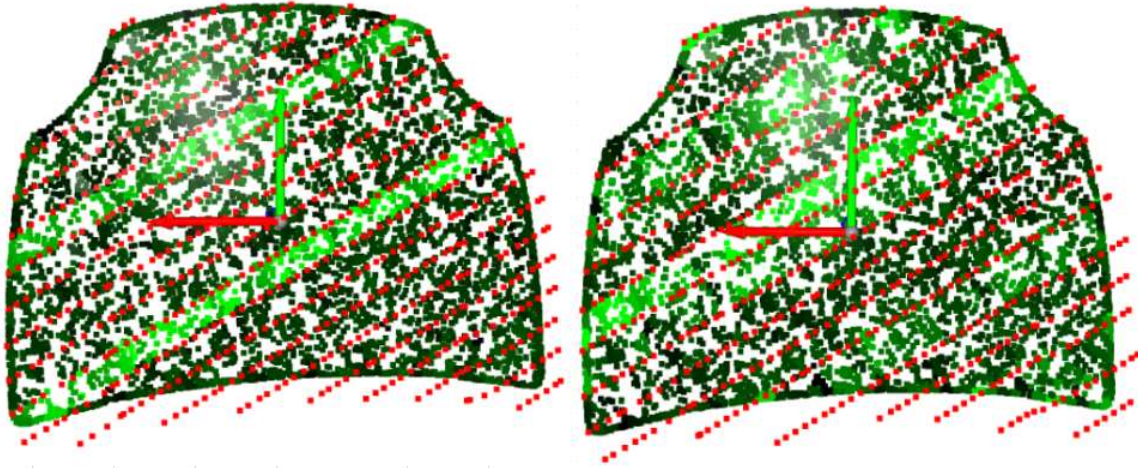
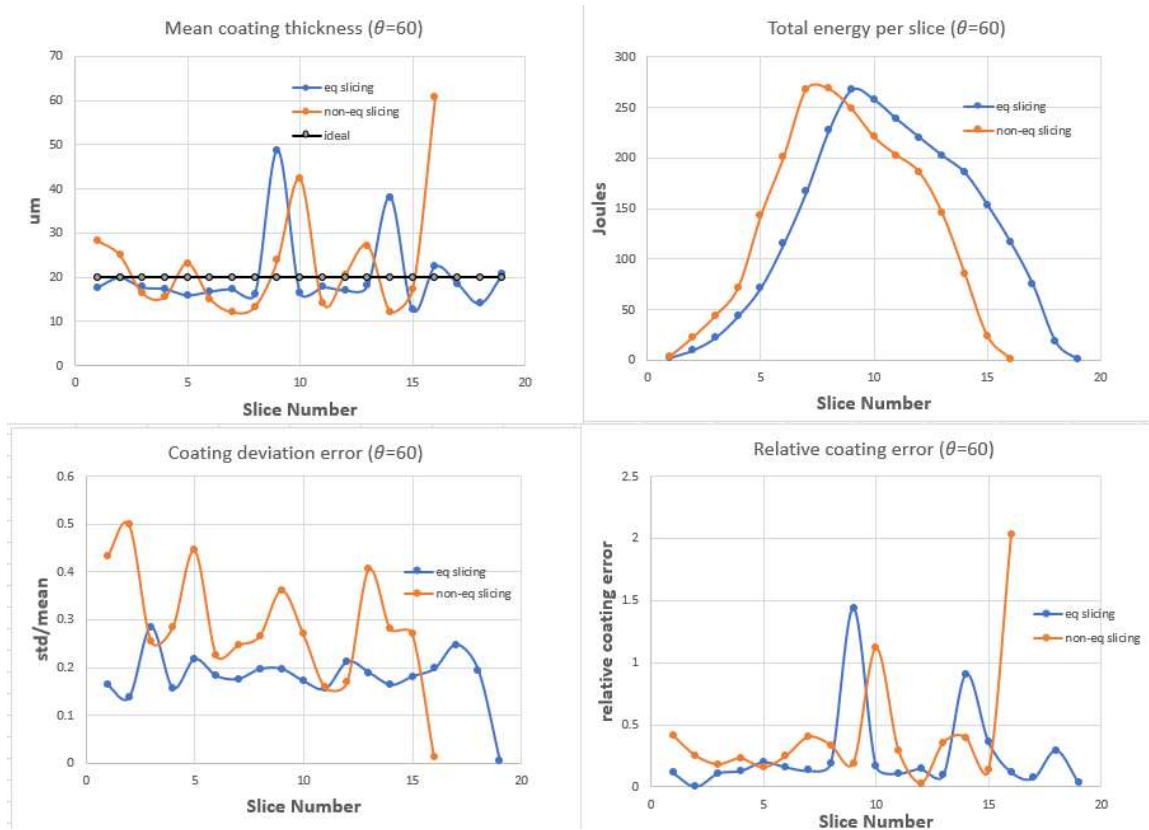


Figure 6.20: Coating thickness and planned trajectory of a car hood in {EF} for slicing direction  $\theta = 60^\circ$  (left: equidistant slicing, right: non-equidistant slicing).



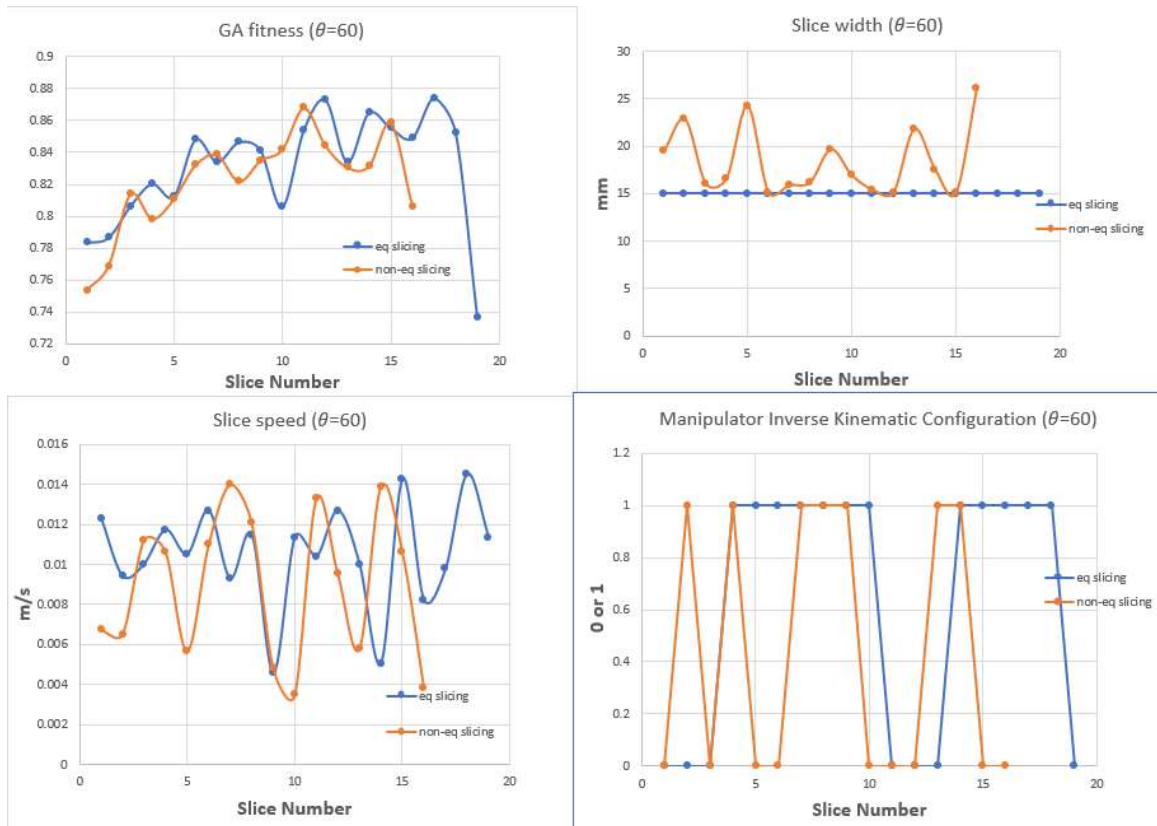


Figure 6.21: Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car hood:  $\theta = 60^\circ$ ).

#### 6.4.4 Results for slicing direction $\theta = 90^\circ$

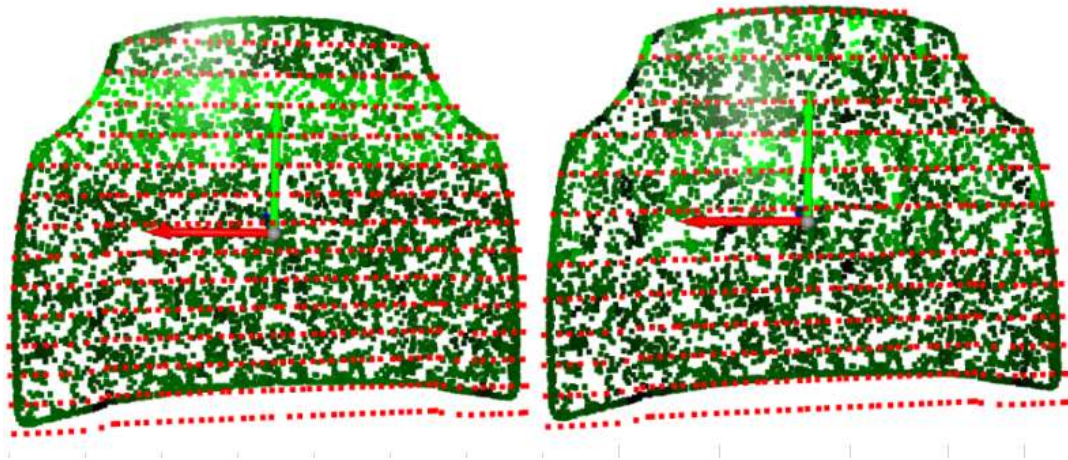


Figure 6.22: Coating thickness and planned trajectory of a car hood in {EF} for slicing direction  $\theta = 90^\circ$  (left: equidistant slicing, right: non-equidistant slicing).



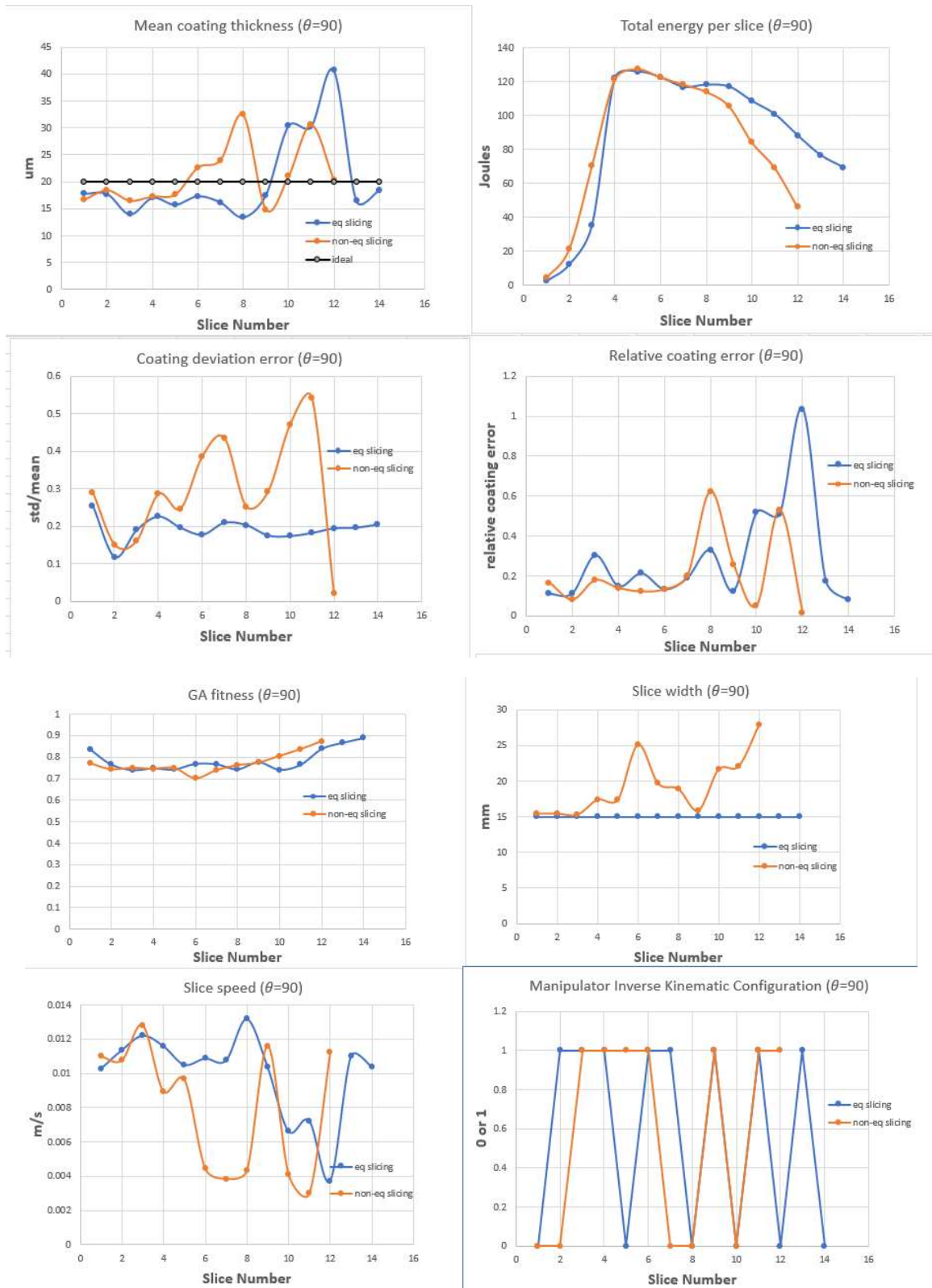


Figure 6.23: Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car hood:  $\theta = 90^\circ$ ).

## 6.4.5 Results discussions

The results for the car hood indicate that a desired coating thickness on a surface can be achieved by specifying the correct spraying parameters and the robot model. The most energy-efficient trajectory is obtained at a slicing direction of  $90^\circ$  and by employing a non-equidistant slicing scheme as shown in Fig. 6.24. It consumes a total of 1027 J of energy which is 73% lower than the least energy-efficient trajectory. The best time-optimal trajectory is observed at an angle of  $30^\circ$  with equidistant slicing scheme. This leads to a total trajectory time of 357 s. The coating deviation error achieved using equidistant slicing is generally lower than non-equidistant slicing scheme consistent with the results for car door. The lowest coating deviation is 18% and is observed at an angle of  $60^\circ$ . The smallest relative coating error (21%) is observed at  $90^\circ$  while employing a non-equidistant scheme. This leads to a mean coating thickness of  $21.02 \mu\text{m}$  over the surface of the car hood. A summary of the results is described in Table 6.5.

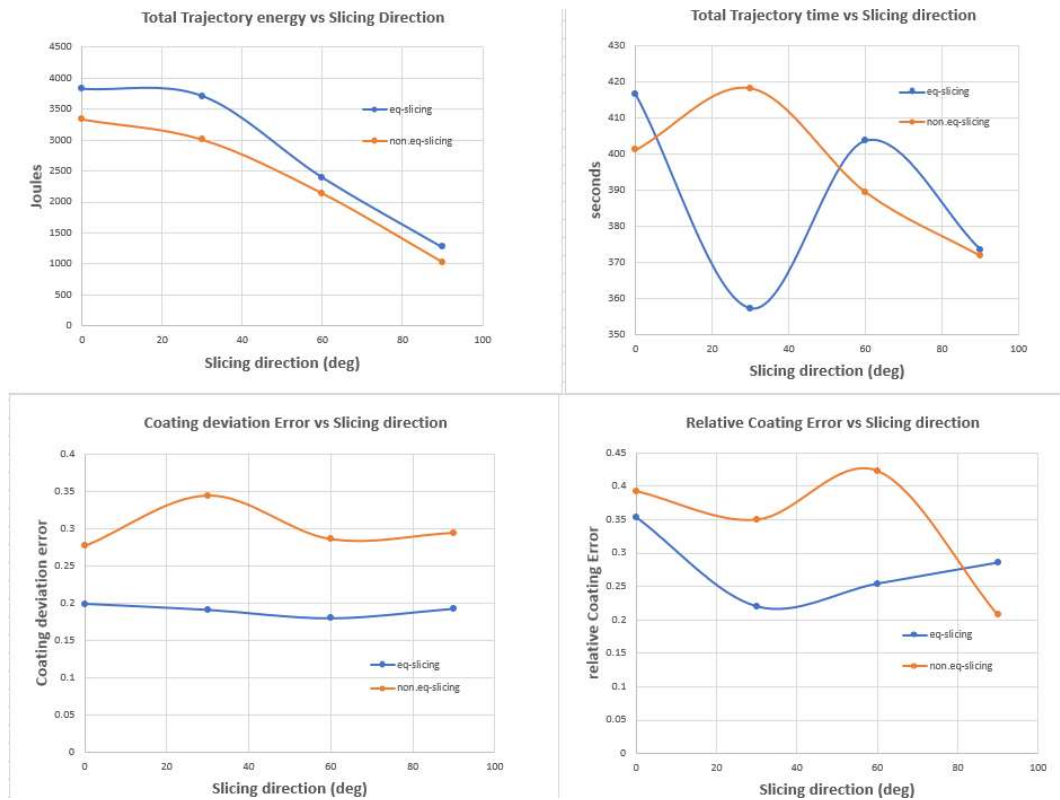


Figure 6.24: Total energy, trajectory time, coating deviation, and relative coating error vs slicing direction (car hood).

Table 6.5: Results summary for trajectory planning and optimization of a car hood for both equidistant and non-equidistant slicing.

	$\theta$	$d_{mean}(\mu m)$	$E_{sum} (J)$	$T_{sum} (s)$	$J_{d_{error}}$	$J_{d_{rel}}$
Eq-slicing	$0^\circ$	22.59	3826.96	416.37	0.20	0.35
	$30^\circ$	19.79	3712.32	357.27	0.19	0.22
	$60^\circ$	20.20	2397.28	403.69	0.18	0.25
	$90^\circ$	20.20	1272.48	373.41	0.19	0.29
Non-eq slicing	$0^\circ$	24.34	3332.79	401.12	0.28	0.39
	$30^\circ$	22.32	3006.22	418.16	0.34	0.35
	$60^\circ$	22.91	2137.09	389.37	0.29	0.42
	$90^\circ$	21.02	1027.1	371.74	0.29	0.21

## 6.5 Optimal paint trajectory planning for a car bumper

### 6.5.1 Results for slicing direction $\theta = 0^\circ$

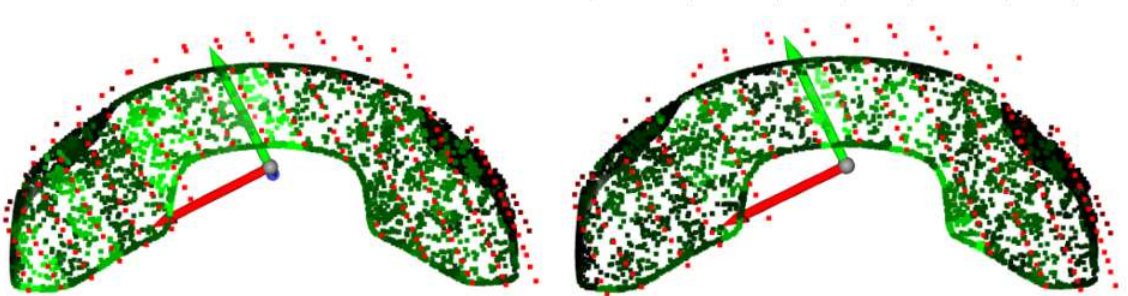
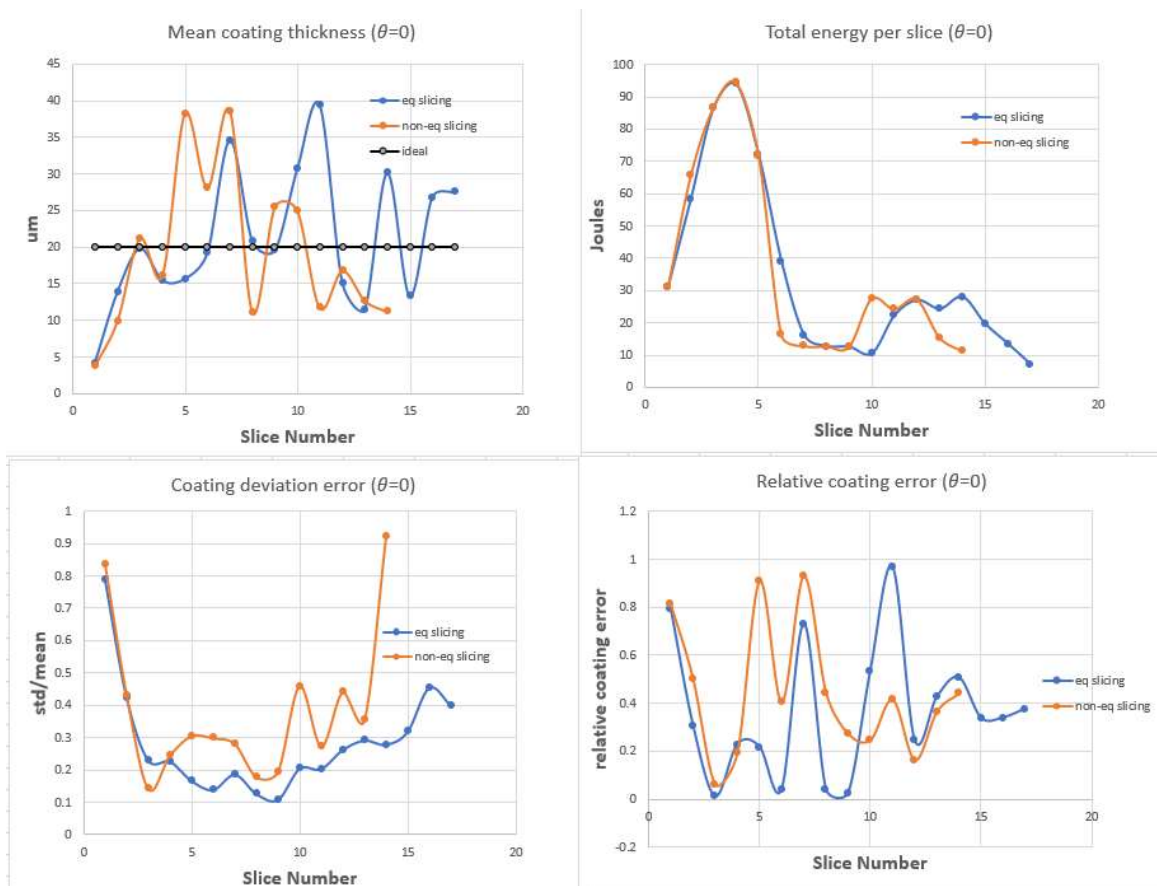


Figure 6.25: Coating thickness and planned trajectory of a car bumper in {EF} for slicing direction  $\theta = 0^\circ$  (left: equidistant slicing, right: non-equidistant slicing).



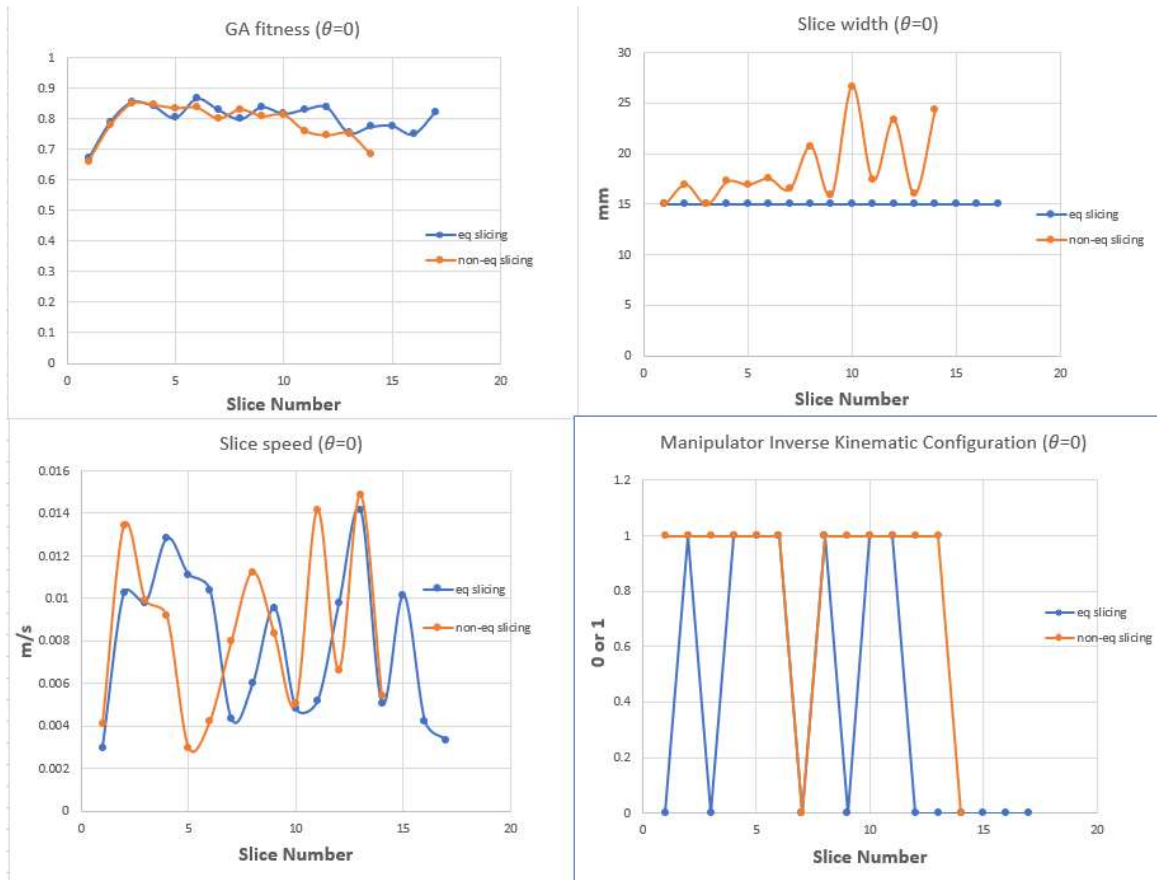


Figure 6.26: Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car bumper:  $\theta = 0^\circ$ ).

### 6.5.2 Results for slicing direction $\theta = 30^\circ$

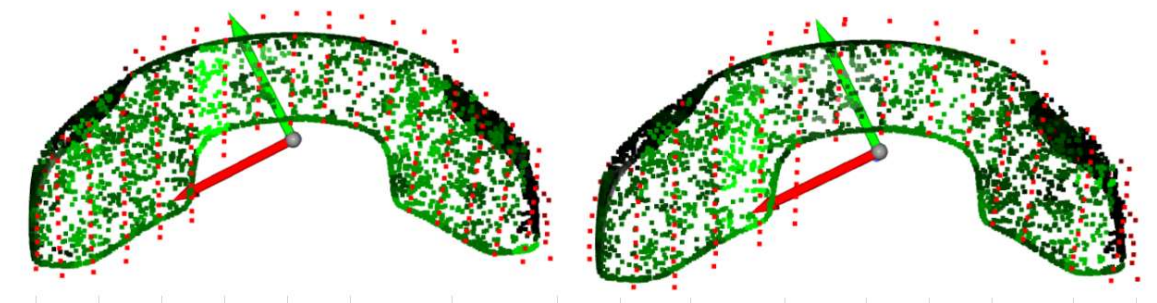


Figure 6.27: Coating thickness and planned trajectory of a car bumper in  $\{EF\}$  for slicing direction  $\theta = 30^\circ$  (left: equidistant slicing, right: non-equidistant slicing).

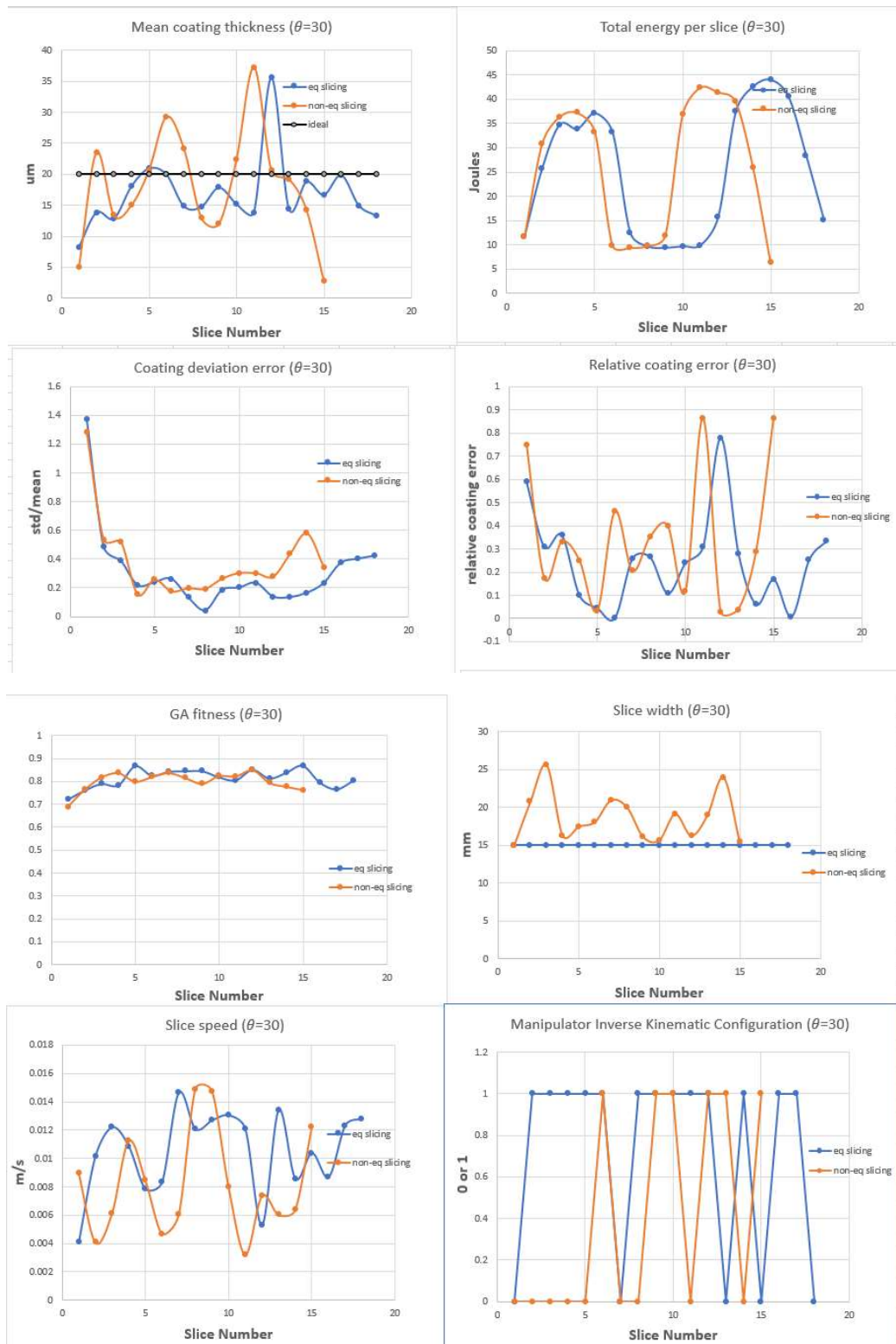


Figure 6.28: Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car bumper:  $\theta = 30^\circ$ ).



### 6.5.3 Results for slicing direction $\theta = 60^\circ$

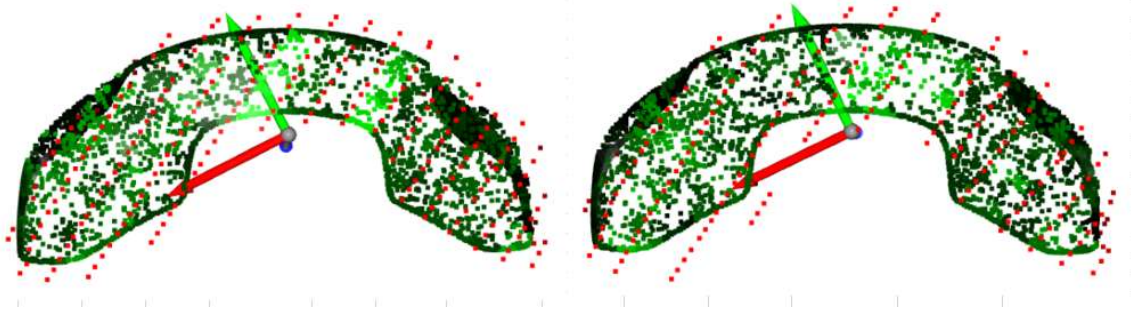
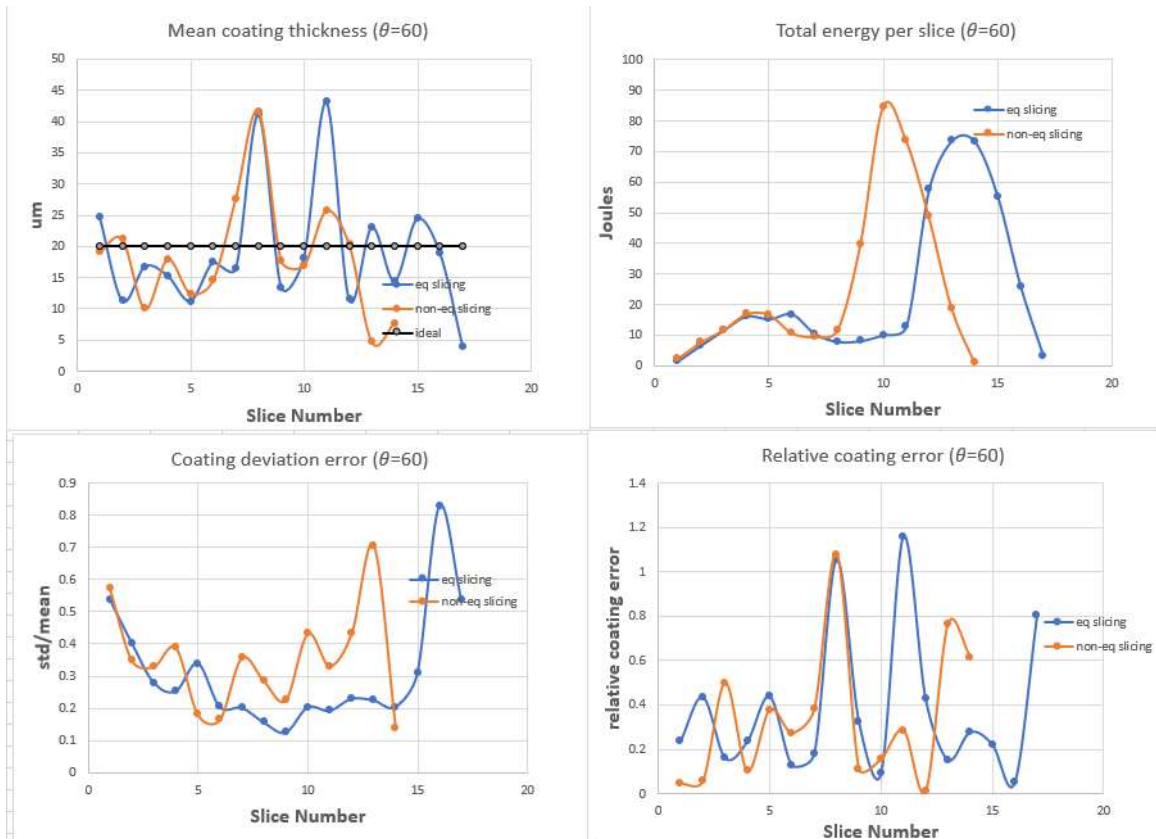


Figure 6.29: Coating thickness and planned trajectory of a car bumper in {EF} for slicing direction  $\theta = 60^\circ$  (left: equidistant slicing, right: non-equidistant slicing).



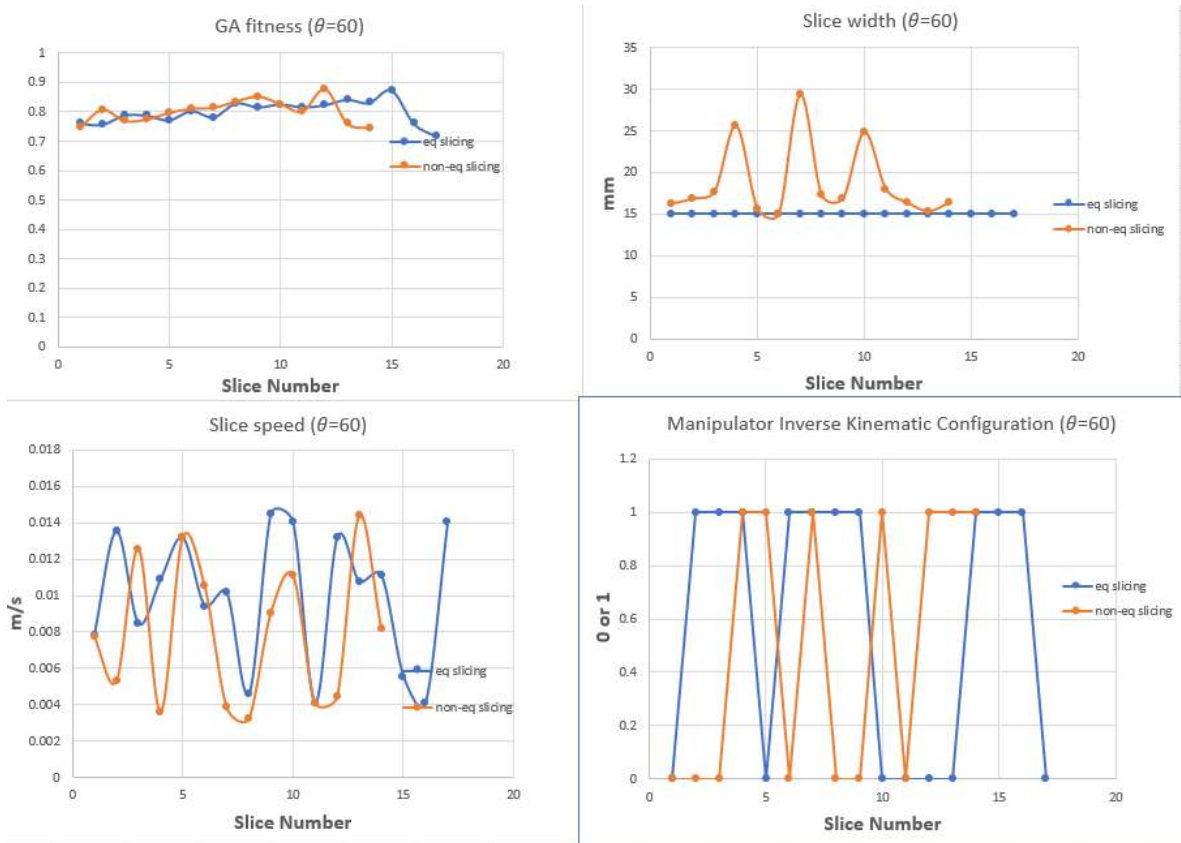


Figure 6.30: Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car bumper:  $\theta = 60^\circ$ ).

#### 6.5.4 Results for slicing direction $\theta = 90^\circ$

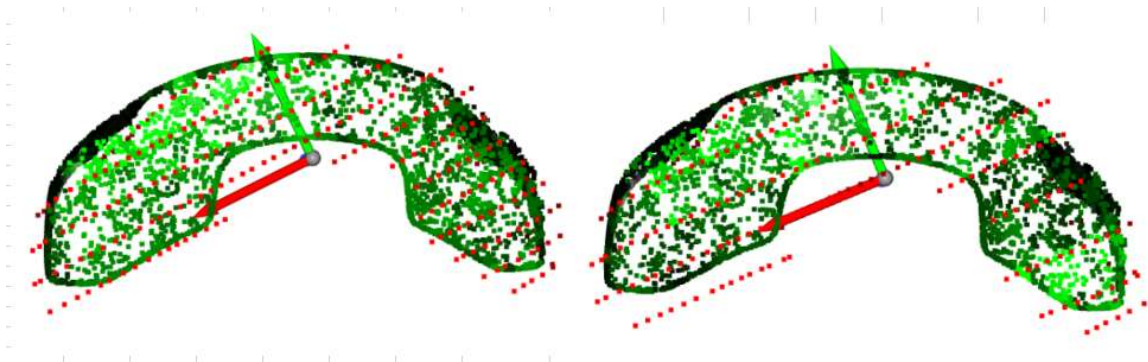


Figure 6.31: Coating thickness and planned trajectory of a car bumper in  $\{EF\}$  for slicing direction  $\theta = 90^\circ$  (left: equidistant slicing, right: non-equidistant slicing).



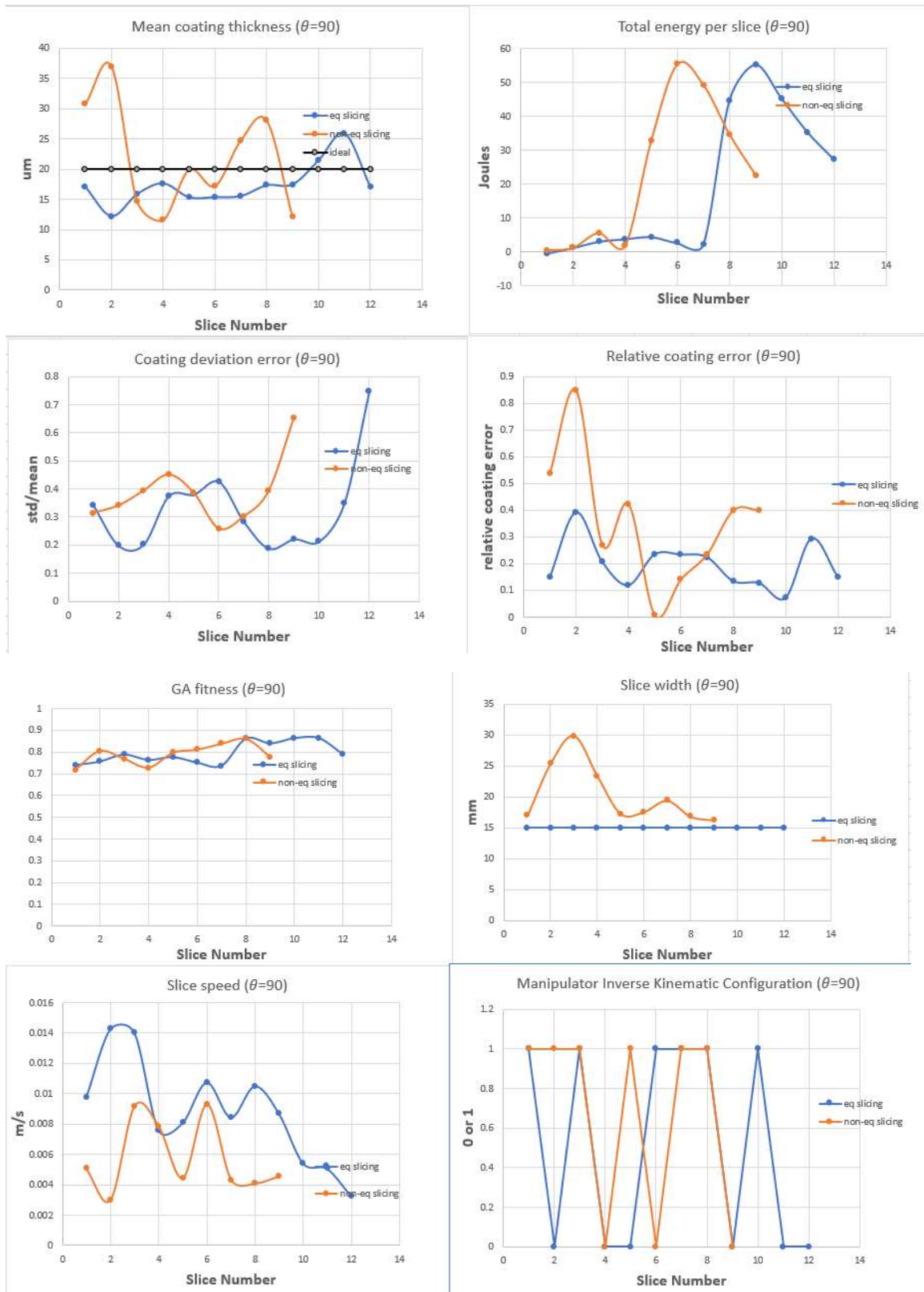


Figure 6.32: Mean coating thickness, energy, coating deviation and relative coating error, fitness, slice width, speed, and inverse kinematic configuration (car bumper:  $\theta = 90^\circ$ ).

### 6.5.5 Results discussions

The results for the car bumper indicate that a desired coating thickness on a surface can be achieved by specifying the correct spraying parameters and the robot model. The most energy-efficient trajectory is obtained at a slicing direction of  $90^\circ$  for non-equidistant scheme as shown in Fig. 6.33. The total energy savings are 64%. The energy consumption decreases with the increase in the slicing direction and is consistent with the results obtained for the car door and car hood. The lowest trajectory time is 137 s when using equidistant slicing at an angle of  $30^\circ$ . Similarly, consistent with the results of car door and hood, the coating deviation error is lower for equidistant slicing scheme with the smallest value (28%) observed at an angle of  $0^\circ$  which gives a mean coating thickness of  $21.02 \mu\text{m}$ . The relative coating error follows a similar trend with an exception at an angle of  $60^\circ$ . The smallest relative coating error is 19% and is observed for equidistant slicing at an angle of  $90^\circ$ . A summary of the results for the car bumper is given in Table 6.6.

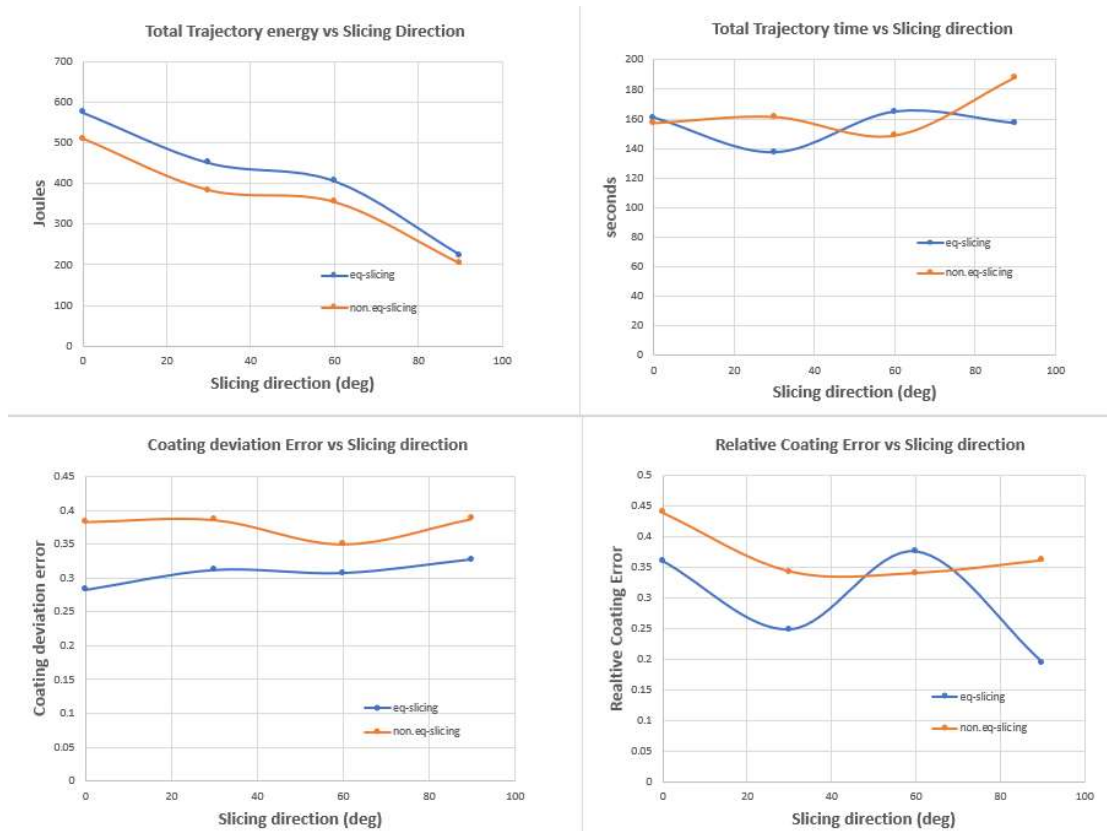


Figure 6.33: Total energy, trajectory time, coating deviation and relative coating error vs slicing direction (car bumper).

Table 6.6: Results summary for trajectory planning and optimization of a car bumper for both equidistant and non-equidistant slicing.

	$\theta$	$d_{mean}(\mu m)$	$E_{sum} (J)$	$T_{sum} (s)$	$J_{d_{error}}$	$J_{d_{rel}}$
Eq-slicing	$0^\circ$	21.02	575.59	160.58	0.28	0.36
	$30^\circ$	16.86	451.24	137.47	0.31	0.25
	$60^\circ$	19.13	406.39	164.68	0.31	0.38
	$90^\circ$	17.33	223.84	156.80	0.33	0.19
Non-eq slicing	$0^\circ$	19.27	509.64	157.04	0.38	0.44
	$30^\circ$	18.15	383.29	161.19	0.39	0.34
	$60^\circ$	18.38	354.33	148.73	0.35	0.34
	$90^\circ$	21.74	203.25	188.07	0.39	0.36

## 6.6 Experimental validation of energy consumption

To assess the validity of the optimization results, the trajectory is executed in real time on a PRRR manipulator defined earlier. A current sensor is installed on the power inlet of the robot to measure the power drawn as the trajectory is being executed. The idea is to compare the energy consumption of the least and most energy-efficient trajectories. For the car door, the trajectories with slicing direction of  $30^\circ$  and  $90^\circ$  with equidistant and non-equidistant slicing schemes are selected respectively. The current measurements at a given trajectory point and the time duration between consecutive trajectory points are logged. When the energy values are summed across the trajectory points, it is observed that the non-equidistant slicing leads to energy savings of 44%. This value is close to the theoretical estimation of 60%. In practice, some of the energy is lost by overcoming the friction between link joints and the resistance in the electrical circuit. The real-time energy values are plotted against the trajectory points as shown in Fig. 6.34.

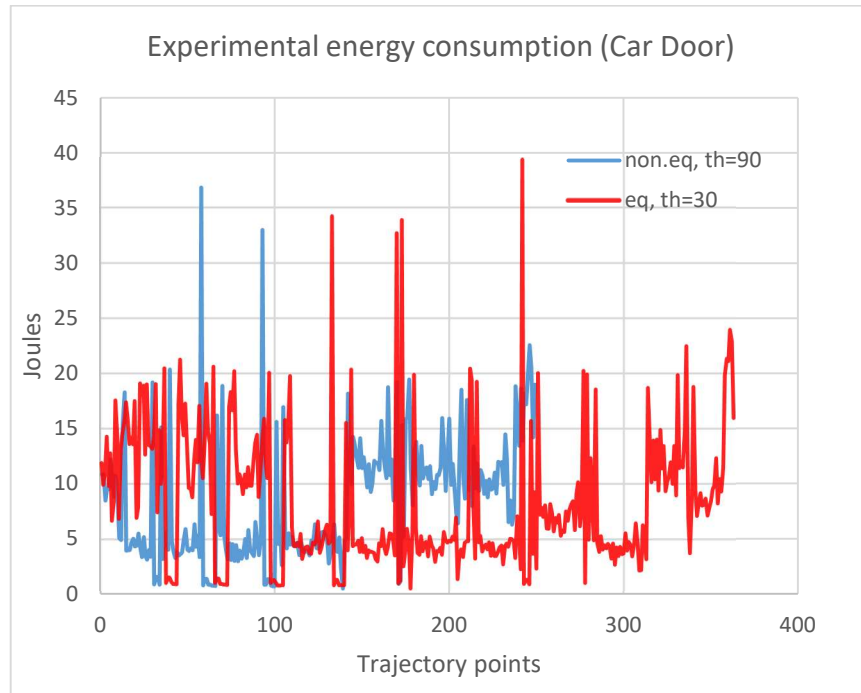


Figure 6.34: Experimental energy consumption for trajectory optimization of car door.

Similarly, the experimental validation of the energy consumption for the car hood shows a total energy savings of 51% if non-equidistant slicing is used with a slicing direction of  $90^\circ$ . It is compared with equidistant slicing at a slicing direction of  $0^\circ$  as shown in Fig. 6.35.

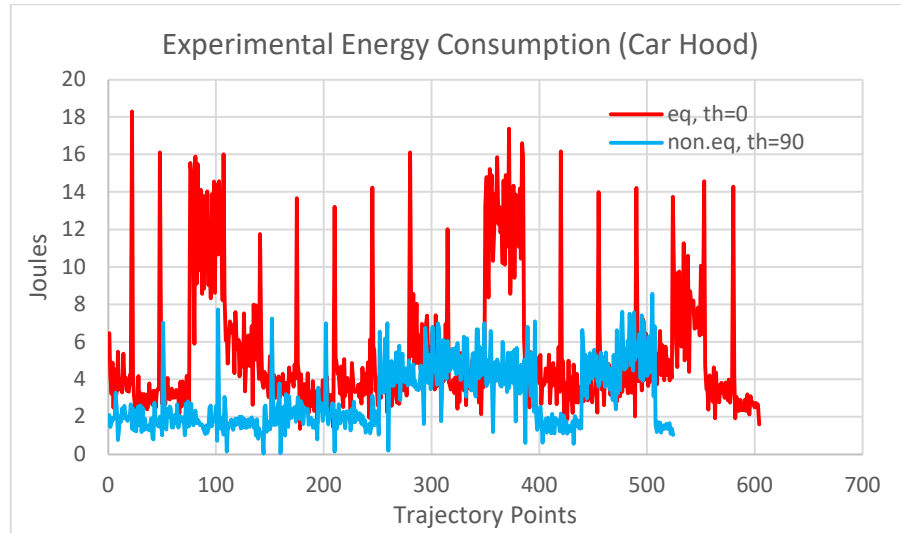


Figure 6.35: Experimental energy consumption for trajectory optimization of car hood.

Finally, the experimental validation of the energy consumption for the car bumper shows a total energy savings of 33% if non-equidistant slicing is used with a slicing direction of  $90^\circ$ . It is compared with equidistant slicing at a slicing direction of  $0^\circ$  as shown in Fig. 6.36.

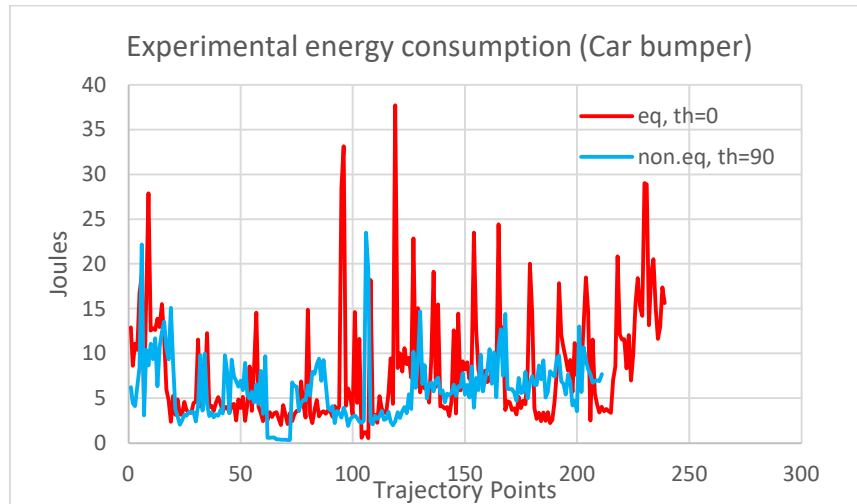


Figure 6.36: Experimental energy consumption for trajectory optimization of car bumper.



Figure 6.37: Robot executing trajectory on a car door.

A summary of theoretical and experimental results is given in Table 6.7. Total trajectory energy is  $E_{sum}$ , while  $E_{sav}$  represents the percentage of energy savings compared to the reference trajectory.

Table 6.7: Results summary of experimental validation of energy consumption for optimal paint trajectories of car door, car hood and car bumper.

	Slicing Scheme	$\theta$	Experimental $E_{sum}$	Theoretical $E_{sav}$	Experimental $E_{sav}$
<b>Car door</b>	Non-equidistant	90°	2085 J	<b>60%</b>	<b>44%</b>
	Equidistant (Reference)	30°	3003 J	0%	0%
<b>Car hood</b>	Non-equidistant	90°	1569 J	<b>73%</b>	<b>51%</b>
	Equidistant (Reference)	0°	3212 J	0%	0%
<b>Car bumper</b>	Non-equidistant	90°	1275 J	<b>64%</b>	<b>33%</b>
	Equidistant (Reference)	0°	1894 J	0%	0%

## 6.7 Results comparison with literature

The coating uniformity, trajectory time, and energy consumption in paint application are dependent on the geometry of the object, dynamics of the spraying process and the robot performing the trajectory. These will vary based on the scenario presented in the optimization process. It can be concluded from the analysis of the results that the proposed hybrid optimization scheme is able to generate efficient trajectories for the painting process. Some trajectories are efficient in terms of coating quality, others lead to lower process times, while some are more energy efficient. Thus, a suitable trajectory must be selected based on the requirements of the painting. The summary of the results comparison is tabulated in Table 6.8.

Table 6.8: Results comparison summary with literature.

Article	U-direction [36]	V-direction [36]	Equidistant slicing [37]	Non. eq slicing [37]	Transitional-seg opt [38]	Proposed scheme	Proposed scheme	Proposed scheme
Object of interest	Oval Bucket	Oval Bucket	Motorcycle spoiler	Motorcycle spoiler	Aircraft wing	Car door	Car hood	Car bumper
Desired coating thickness	50 $\mu\text{m}$	50 $\mu\text{m}$	23 $\mu\text{m}$	23 $\mu\text{m}$	70 $\mu\text{m}$	20 $\mu\text{m}$	20 $\mu\text{m}$	20 $\mu\text{m}$
Mean Coating thickness	51.1 $\mu\text{m}$	52.2 $\mu\text{m}$	25.95 $\mu\text{m}$	22.27 $\mu\text{m}$	68.7 $\mu\text{m}$	19.21 $\mu\text{m}$	21.02 $\mu\text{m}$	21.02 $\mu\text{m}$
Standard deviation	2.775 $\mu\text{m}$	3.8 $\mu\text{m}$	6.52 $\mu\text{m}$	5.71 $\mu\text{m}$	3.2 $\mu\text{m}$	3.41 $\mu\text{m}$	6.51 $\mu\text{m}$	5.35 $\mu\text{m}$
Mean coating deviation error	5.43%	7.60%	25.12%	25.64%	4.60%	17.75%	29.4%	25.4%
Mean relative coating error	2.2%	4.4%	12.83%	3.17%	1.86%	3.95%	5.1%	5.1%
Max Time savings	17%	0%	N/A	N/A	N/A	33%	14.18%	27.13%
Max Energy savings	N/A	N/A	N/A	N/A	N/A	60%	73%	64%
Coating mean-squared error cost	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Coating deviation cost	No	No	No	No	No	Yes	Yes	Yes
Energy cost	No	No	No	No	No	Yes	Yes	Yes
Process time cost	No	No	No	No	No	Yes	Yes	Yes

The methods and optimization schemes in the literature do not consider process time optimization, coating deviation error, and energy consumption in the cost functions. Our novel hybrid optimization scheme introduces these terms into the objective function, which then leads to more optimal paint trajectories. The performance of our results is in close agreement with the literature analysis as indicated by the proximity of the mean coating thickness with the desired requirement, and the standard deviation indicating the spread in the distribution of coating over the complex surfaces. These results are presented as a summary and should not be used as a direct comparison since the performance indices highly depend on the geometry of the complex surfaces, the spraying process, and the robot model in general.



## Chapter 7. Conclusion and Future Works

The primary goal of the thesis was to develop an integrated system for industrial painting capable of generating automated paint trajectories optimized for coating thickness, process times, and energy consumption. The literature review highlighted key technologies needed to automate the trajectory planning process of a paint robot. The establishment of a coating deposition model on a complex free-form surface and the dynamic model of the robot bears key importance in the trajectory planning and optimization process. While adequate work has been done to optimize the coating thickness over a complex free-form surface, the application of a robot dynamic model to obtain energy efficient paint trajectories is left unattended. Therefore, this study focused on developing a hybrid optimization technique to ascertain coating uniformity, process times, and energy consumption in the trajectory planning process. Moreover, considerable effort was put into the development of the integrated system, specifically the web-based GUI and the backend programming for interacting with the system.

The trajectory planning process starts with the acquisition of the 3D model of the object. Once the geometry of the object is obtained and calibrated in the camera frame, an improved point cloud slicing technique is applied with the provision of a variable slicing direction to broaden the optimizer search space. The slicing is performed at an arbitrary angle of the eigen coordinate frame. Then, the trajectory points for each slice are obtained in discrete steps, and the coating thickness is computed. Similarly, using the robot dynamic model, the joint torques, link velocities, and the time delta between trajectory points are computed. The individual slice of the point cloud is then optimized using GA (Genetic Algorithm) by minimizing a joint cost function. The inclusion of slicing direction and the inverse kinematic configuration of the robot among the slice width and slice speeds leads to a better optimization of the trajectories in terms of coating uniformity, process times, and energy consumption. Using the hybrid optimization scheme and employing a variable slicing direction, optimal paint trajectories can be obtained. The literature analysis revealed the use of only a single mean squared error objective function to achieve the desired coating thickness requirement. Our novel optimization scheme introduced three additional terms

into the objective function to account for the coating deviation errors, process times, and energy consumption. Experimental results reveal energy savings of **44%**, **51%** and **33%** for the paint trajectories of a car door, car hood, and car bumper, respectively, while achieving coating uniformity and lower process times.

For future works, the paint system can be modeled in a simulation environment like Gazebo [82] or MATLAB [83] to analyze the optimized paint trajectories. Gazebo is a physics engine and requires the model to be presented in the form of a URDF (Unified Robot Description Format), which works in conjunction with the ROS ecosystem. A URDF can model the kinematic and dynamic properties of the robot in more detail, leading to more accurate calculations of the robot dynamics. On a similar note, the spray delivery system can also be modeled, and the coating thickness can be analyzed. The use of a simulation environment makes it easy to assess the performance of the system while eliminating the need for a physical system.

Another recommendation for future work would be the use of large 6-axis robots with the provision of an HVLP spray gun system. The last 3 axes of the robot can be used for orientation control to position the paint gun over the surface of the object. The experimental paint application is not possible with a 4 DOF robot since it can only be controlled for a position in the three-dimensional cartesian space and not the orientation. In the literature review section, industrial robotic systems for paint applications were discussed in detail and can be utilized for paint applications more precisely.

Nevertheless, for industrial applications, where many parts are to be processed simultaneously, the hybrid optimization scheme can be implemented on a large scale. The optimization function implemented in Python can be converted to a class object, and hyperthreading can be utilized to parallelize the execution. A class instance for each object can be submitted to a standalone thread, thereby making the execution parallel. Moreover, GPU processing can also be investigated to process large batch sizes of objects, and the trajectories can be computed. As a final recommendation, the execution scripts can be built using docker containers [84] with the required dependencies included on a single image. The proposed solution will lead to a smoother distribution of the software resources independent of the OS used by the server.

## References

- [1] "Global automotive executive survey Tech. Rep.," KPMG, 2017.
- [2] "Volatile Organic Compounds' Impact on Indoor Air Quality," United States Environmental Protection Agency, [Online]. Available: <https://shorturl.at/ekpR4..>
- [3] "Gone to waste: exploring the environmental consequences of industrial paint pollution," qlayers, [Online]. Available: <https://shorturl.at/CGS16>.
- [4] G. J. Bambosek, Algonac, D. S. Bartlett and T. D. Schmidt, "SPRAY PAINT SYSTEM INCLUDING PAINT BOOTH, PAINT ROBOT APPARATUS MOVABLE THEREIN AND RAIL MECHANISM FOR SUPPORTING THE APPARATUS THEREOUT". US Patent 4,630,567, 23 Dec 1986.
- [5] S.-S. Suh, J.-J. Lee, Y.-J. Choi and S.-K. Lee, "A Prototype Integrated Robotic Painting System: Software and Hardware Development," in *Proceedings of the 1993 IEE4RSJ International Conference on Intelligent Robots and Systems*, Yokohama, Japan, 1993.
- [6] M. A. S. Arıkan and T. Balkan, "Process Modeling, Simulation, and Paint Thickness Measurement for Robotic Spray Painting," *Journal of Field Robotics*, vol. 17, no. 9, pp. 479-494, 2000.
- [7] "Model 95A & 95AR Automatic Spray Gun," Carlisleleft, [Online]. Available: <https://carlisleleft.com/en/product/model-95-automatic-spray-gun/>.
- [8] Elcometer, "Elcometer 345 Coating Thickness Gauge," [Online]. Available: [https://www.mltest.com/images/stories/elco\\_345.pdf](https://www.mltest.com/images/stories/elco_345.pdf).
- [9] "Iwata Revolution HP-M2 Gravity Feed Single Action Airbrush," Iwata, [Online]. Available: <https://www.iwata-airbrush.com/revolution-mini-hp-m2.html>.
- [10] L. Scalera, E. Mazzon, P. Gallina and A. Gasparetto, "Airbrush Robotic Painting System: Experimental Validation of a Colour Spray Model," in *International Conference on Robotics in Alpe-Adria Danube Region: Advances in Service and Industrial Robotics*, Danube, July 2017.
- [11] M. Javaid, A. Haleem, S. Pratap and R. Suman, "Industrial perspectives of 3D scanning: Features, roles and it's analytical applications," *Sensors International* , vol. 2, 2021.
- [12] L. Du, Y. Lai, C. Luo, Y. Zhang, J. Zheng, X. Ge and Y. Liu, "E-quality Control in Dental Metal Additive Manufacturing Inspection Using 3D Scanning and 3D Measurement," *Frontiers in Bioengineering and Biotechnology*, vol. 8, 2020.

- [13] C. E Dombroski, M. ER Balsdon and A. Froats, "The use of a low cost 3D scanning and printing tool in the manufacture of custom-made foot orthoses: a preliminary study," *BMC Research Notes*, London, ON, Canada, Dec 2014.
- [14] W. L. Y. A, "Applications of 3D scanning and reverse engineering techniques for quality control of quick response products," *Int J Adv Manuf Technol*, vol. 26, p. 1284–1288, 2005.
- [15] W. K. K, D. P. M, E. C. M, B. P, L. C and B. R, "Uncertainty studies of topographical measurements on steel surface corrosion by 3D scanning electron microscopy," *Micron* 43, pp. 387-395, 2012.
- [16] J.-F. Larue, D. Brown and M. Viala, "How optical CMMs and 3D scanning will Revolutionize the 3D Metrology World," in *Integrated Imaging and Vision Techniques for Industrial Inspection*, London, Springer, 2015, pp. 141-176.
- [17] O. Esaias, G. W. Noonan, S. Everist, M. Roberts, C. Thompson and M. N. Krosch, "Improved Area of Origin Estimation for Bloodstain Pattern Analysis Using 3D Scanning," *J Forensic Sci*, vol. 65, no. 3, pp. 722-728, May 2020.
- [18] K. Panjvani, A. V. Dinh and K. A. Wahid, "LiDARPheno – A Low-Cost LiDAR-Based 3D Scanning System for Leaf Morphological Trait Extraction," *frontiers in Plant Science*, vol. 10, Feb 2019.
- [19] C. Little, D. Patterson, B. Moyle and A. Bec, "Every footprint tells a story: 3D scanning of heritage artifacts as an interactive experience," in *Proceedings of the Australasian Computer Science Week Multiconference*, 2018.
- [20] Z. Stojkic, E. Culjak and L. Saravanja, "3D MEASUREMENT - COMPARISON OF CMM AND 3D SCANNER," in *31ST DAAAM INTERNATIONAL SYMPOSIUM ON INTELLIGENT MANUFACTURING AND AUTOMATION*, 2020.
- [21] O. Alexandrov and R. A. Beyer, "Multiview Shape-From-Shading for Planetary Images," *Eath and Space Science*, vol. 5, pp. 652-666, 2018.
- [22] N. Ye, H. Zhu, M. Wei and L. Zhang, "Accurate and dense point cloud generation for industrial Measurement via target-free photogrammetry," *Optics and Lasers in Engineering*, vol. 140, 2021.
- [23] R. A. Newcombe, S. Izadi, O. Hilliges and D. Molyneaux, "KinectFusion: Real-Time Dense Surface Mapping and Tracking," in *10th IEEE International Symposium on Mixed and Augmented Reality*, Basel, Switzerland, 2011.

- [24] S. Meister, S. Izadi, P. Kohli, M. Hammerle, C. Rother and D. Kondermann, "When Can We Use KinectFusion for Ground Truth Acquisition?," in *Proc. Workshop on Color-Depth Camera Fusion in Robotics*, 2012.
- [25] F. Ma, L. Carlone, U. Ayaz and S. Karaman, "Sparse depth sensing for resource-constrained robots," *The International Journal of Robotics Research.*, vol. 38, no. 8, pp. 935-980, 2019.
- [26] C. Boehnen and P. Flynn, "Accuracy of 3D Scanning Technologies in a Face Scanning Scenario," in *Fifth International Conference on 3-D Digital Imaging and Modeling*, Ottawa, ON, Canada, 2005.
- [27] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vols. 14., no. 2, p. 239–256, Feb, 1992.
- [28] Y. Lei, M. Bennamoun, M. Hayat and Y. Guo, "An efficient 3D face recognition approach using local geometrical signatures," *Pattern Recognition*, vol. 47, no. 2, pp. 509-524, February 2014.
- [29] R. Osada, T. Funkhouser, B. Chazelle and D. Dobkin, "Matching 3D models with shape distributions," in *Proceedings International Conference on Shape Modeling and Applications*, Genova, Italy, pp. 154-166, 2001.
- [30] S. Larsson and J. A. . Kjellander, "Motion control and data capturing for laser scanning with an industrial robot," *Robotics and Autonomous Systems*, vol. 54 , p. 453–460, 2006.
- [31] T. Borangiu and A. Dumitrache, Robot arms with 3D vision capabilities, Romania, 2010.
- [32] J. Li, M. Chen, X. Jin, Y. Chen, Z. Dai, Z. Ou and Q. Tang, "Calibration of a multiple axes 3-D laser scanning system consisting of robot, portable laser scanner and turntable," *Optik*, vol. 122, no. 4, pp. 324-329, February 2011.
- [33] C. Shen and S. Zhu, "A Robotic System for Surface Measurement Via 3D Laser Scanner," in *The 2nd International Conference on Computer Application and System Modeling*, 2012.
- [34] A. Pichler, h. Viicze, H. Andersen and O. hladseii, "A Method for Automatic Spray Painting of Unknown Parts," in *International Conference on Robotics & Automation*, Washington, DC, May 2002.
- [35] M. V. Andulkar and S. S. Chiddarwar, "Incremental approach for trajectory generation of spray painting robot," *Industrial Robot: An International Journal*, vol. 42, no. 3, pp. 228-241, 2015.

- [36] W. Chen, Y. Tang and Q. Zhao, "A novel trajectory planning scheme for spray painting robot with Bézier curves," in *Chinese Control and Decision Conference (CCDC)*, Yinchuan, China, 6746-6750, 2016.
- [37] X. Yu, Z. Cheng, Y. Zhang and L. Ou, "Point cloud modeling and slicing algorithm for trajectory planning of spray painting robot," *Robotica*, vol. 39, p. 2246–2267, 2021.
- [38] L. Guan and L. Chen, "Trajectory planning method based on transitional segment optimization of spray transitional segment optimization of spray," *Industrial Robot: the international journal of robotics research and application*, vol. 46, no. 1, pp. 31-43, 2019.
- [39] G. L. Srinivas and A. Javed, "Optimization approaches of industrial serial manipulators to improve energy efficiency: A review," in *3rd International Conference on Advances in Mechanical Engineering (ICAME)*, 2020.
- [40] K. Paes, W. Dewulf, K. V. Elst, K. Kellens and P. Slaets, "Energy efficient trajectories for an industrial ABB robot," in *21st CIRP Conference on Life Cycle Engineering*, 2014.
- [41] A. Sengupta, T. Chakraborti, A. Konar and A. Nagar, "Energy Efficient Trajectory Planning by a Robot Arm using Invasive Weed," in *Third World Congress on Nature and Biologically Inspired Computing*, Salamanca, Spain, 2011.
- [42] A. Fenucci, M. Indri and F. Romanelli, "An off-line robot motion planning approach for the reduction of the energy consumption," in *IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, Berlin, Germany, 2016.
- [43] J. N. Pires, T. Godinho and P. Ferreira, "CAD interface for automatic robot welding programming," *The Industrial Robot*, vol. 31, no. 1, pp. 71-76, 2004.
- [44] H. Chen, T. Fuhlbrigge and X. Li, "A review of CAD-based robot path planning for spray painting.," *Industrial Robot*, vol. 36, no. 1, p. 45–50, 2009.
- [45] N. R. Whitehouse, "Shreir's Corrosion || Paint Application," 2010.
- [46] S. V. Ravikumar, J. M. Jha, I. Sarkar, S. K. Pal and S. Chakrabortya, "Enhancement of heat transfer rate in air-atomized spray cooling of a hot steel plate by using an aqueous solution of non-ionic surfactant and ethanol," *Applied Thermal Engineering*, vol. 64, no. 1–2, pp. 64-75, March 2014.
- [47] S. Poozesh, N. Akafuah and K. Saito, "Effects of automotive paint spray technology on the paint transfer efficiency – a review," *Journal of Automotive Engineering*, vol. 232, no. 2, p. 282–301, 2018.

- [48] J. Casanova, J. Lima and P. Costa, "A Simulation Tool for Optimizing a 3D Spray Painting System," in *International Conference on Optimization, Learning Algorithms and Applications OL2A*, Springer, Cham, Jan 2022.
- [49] J. Rupp, E. Guffey and G. Jacobsen, "Electrostatic spray processes," *Metal Finishing*, vol. 108, no. 11-12, p. 150–163, Dec 2010.
- [50] RoboDK, "Simulate Robot Applications," RoboDK, [Online]. Available: <https://robodk.com/examples%7B#}examples-painting>.
- [51] "Robotic and automated workcell simulation, validation and offline programming," [Online]. Available: [www.geoplms.com/knowledge-base-resources/GEOPLM-Siemens-PLM-Tecnomatix-Robcad.pdf](http://www.geoplms.com/knowledge-base-resources/GEOPLM-Siemens-PLM-Tecnomatix-Robcad.pdf).
- [52] "Delfoi PAINT," Delfoi, [Online]. Available: <https://www.delfoi.com/delfoi-robotics/delfoi-paint/>.
- [53] "RobotStudio® Painting PowerPac," ABB, [Online]. Available: <https://new.abb.com/products/robotics/application-software/painting-software/robotstudio-painting-powerpac>.
- [54] "Inropa™ OLP Automatic," [Online]. Available: [https://www.inropa.com/fileadmin/Arkiv/Dokumenter/Produktblade/OLP\\_automatic.pdf](https://www.inropa.com/fileadmin/Arkiv/Dokumenter/Produktblade/OLP_automatic.pdf).
- [55] "FANUC ROBOGUIDE PAINTPRO," FANUC, [Online]. Available: <https://www.fanucamerica.com/support/training/robot/elearn/fanuc-roboguide-paintpro>.
- [56] MMC, "ATEX Codes Simplified," MeasureMonitorControl, [Online]. Available: <https://www.measuremonitorcontrol.com/resources/atex/atex-codes>.
- [57] KUKA, "KUKA ready2\_spray," KUKA, [Online]. Available: [pdf.directindustry.com/pdf/kuka-ag/kuka-ready2-spray/17587-748199.html](http://pdf.directindustry.com/pdf/kuka-ag/kuka-ready2-spray/17587-748199.html).
- [58] FANUC, "FANUC P-250iB Paint Robot," FANUC, [Online]. Available: <https://www.fanucamerica.com/products/robots/series/paint/p-250ib-paint-robot>.
- [59] ABB, "IRB 5500-22/23," ABB, [Online]. Available: <https://new.abb.com/products/robotics/industrial-robots/irb-5500-22>.
- [60] "Program Compilation (LS vs. TP)," RoboDK, [Online]. Available: <https://robodk.com/doc/en/Robots-Fanuc-Program-Compilation-LS-vs-TP.html>.
- [61] G. v. Rossum, "python," python, [Online]. Available: <https://www.python.org/>.
- [62] "WinOLPC Software," FANUC, [Online]. Available: <https://www.industry-plaza.com/winolpc-software-p27618.html>.

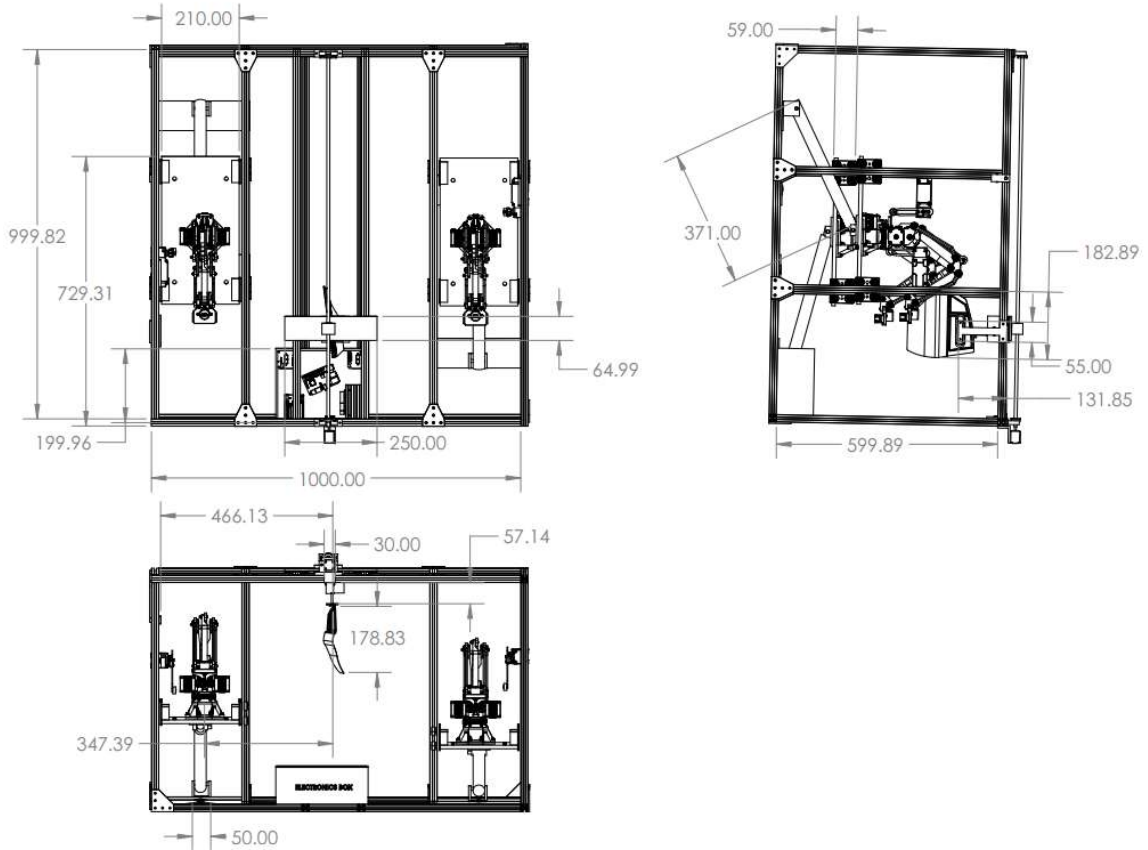
- [63] J. M. Casanova, "Simulation and Planning of a 3D Spray Simulation and Planning of a 3D Spray," Mestrado Integrado em Engenharia Eletrotécnica e de Computadores, July 2021.
- [64] opencv, "Open source computer vision library," <https://github.com/opencv/opencv>.
- [65] "Depth Camera D435," intelrealsense, [Online]. Available: <https://www.intelrealsense.com/depth-camera-d435/>.
- [66] "VL53L0X Time Of Flight Distance Sensor," ESPHome, [Online]. Available: <https://esphome.io/components/sensor/vl53l0x.html>.
- [67] "SG90 Digital," TowerPro, [Online]. Available: <https://www.towerpro.com.tw/product/sg90-7/>.
- [68] E. R. DAVIES, *Machine Vision Theory, Algorithms, Practicalities*, Elsevier Inc, 2005.
- [69] Q.-Y. Zhou, J. Park and V. Koltun, "Open3D: A Modern Library for 3D Data Processing," *Computer Vision and Pattern Recognition*, 2018.
- [70] "From depth map to point cloud," medium, [Online]. Available: <https://medium.com/yodayoda/from-depth-map-to-point-cloud-7473721d3f>.
- [71] J. Xie, Y. Fang, F. Zhu and E. Wong, "DeepShape: Deep Learned Shape Descriptor for 3D Shape Matching and Retrieval," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition CVPR*, pp. 1275-1283, 2015.
- [72] J. J. Craig, "Manipulator Dynamics," in *Introduction to Robotics Mechanics and Control*, Pearson Education International, 2004, pp. 173-176.
- [73] S. Mirjalili, "Genetic Algorithm," in *Evolutionary Algorithms and Neural Networks. Studies in Computational Intelligence, vol 780.*, Cham, Springer, 2019.
- [74] "Nema 23 Stepper Motor Bipolar 1.8 Degree 2.8A," amazon, [Online]. Available: <https://shorturl.at/xyCM6>.
- [75] "Jetmax Jetson nano," Hiwonder, [Online]. Available: <https://www.hiwonder.com/products/jetmax?variant=39645677125719>.
- [76] "Electrical Buddy Adjustable Rod Lever Arm Momentary Limit Switch," amazon, [Online]. Available: <https://www.amazon.ca/Electrical-Buddy-Adjustable-Momentary-Me-8107/dp/B07Y7C9188>.
- [77] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler and A. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, 2009.



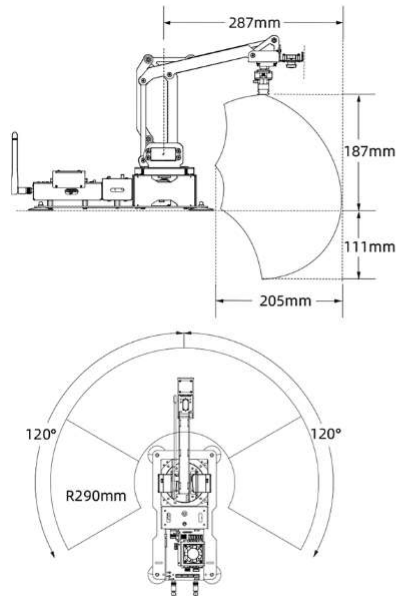
- [78] "JavaScript," Pluralsight, [Online]. Available: <https://www.javascript.com/>.
- [79] "Bootstrap," Bootstrap, [Online]. Available: <https://getbootstrap.com/docs/4.2/getting-started/introduction/>.
- [80] "The Standard ROS JavaScript Library," ROS.org, [Online]. Available: <https://wiki.ros.org/roslibjs>.
- [81] "PyGAD - Python Genetic Algorithm!," pygad, [Online]. Available: <https://pygad.readthedocs.io/en/latest/>.
- [82] "Simulate before you build," GAZEBO, [Online]. Available: <https://gazebo.org/home>.
- [83] "MATLAB," MathWorks, [Online]. Available: <https://www.mathworks.com/products/matlab.html>.
- [84] "Develop faster. Run anywhere.," Docker, [Online]. Available: <https://www.docker.com/>.

## Appendices

### A1. Paint system CAD drawings



### A2. Robot specifications



<b>Axis:</b>	4+1
<b>Dimension:</b>	450*160*260mm
<b>Weight(Only JetMax):</b>	1.6kg
<b>Payload:</b>	Max.450g
<b>Ambient Temperature:</b>	0 °C ~ 45 °C
<b>Material:</b>	Aluminum, Fiberglass
<b>Reach:</b>	R104~R289 x 240°
<b>Repeatability:</b>	±1mm
<b>Joint Speed:</b>	Max. 0.20sec/60°
<b>End Effector:</b>	Suction Cup, Small Gripper, Big Gripper
<b>Power Supply:</b>	12V 5A DC Adapter
<b>Connectivity:</b>	USB/Wi-Fi/Ethernet
<b>Controller:</b>	Jetson Nano B01
<b>Storage:</b>	32G Class 10 TF
<b>Programming Tools:</b>	Python/C/C++/JavaScript
<b>Control Methods:</b>	Computer/ Phone/ Wireless Handle/ Mouse

### A3. Intel Real sense D435 specifications

Metric	D410/D415 (≤ 2 Meters and 80% ROI, HD Resolution)	D430/D435/ D435i/D435f/D435if (≤ 2 Meters and 80% ROI, HD Resolution)	D450/D455/ D455f (≤ 4 Meters and 80% ROI, HD Resolution)	D401/D405 (≤ 0.5 Meters and 80% ROI, HD Resolution)
Z-accuracy (or Absolute Error)	± 2%	± 2%	± 2%	± 2%
Fill rate	≥ 99%	≥ 99%	≥ 99%	≥ 99.5%
RMS Error (or Spatial Noise)	≤ 2%	≤ 2%	≤ 2%	≤ 1%
Temporal Noise	≤ 1%	≤ 1%	≤ 1%	≤ 0.5%
Lifetime	4 years	5 years	5 years	5 years

Depth	Imager	Color (Left Imager) D410	Color (RGB Camera) D415/D435/D435i/ D435f/D435if/ D450/D455/D455f	IMU D435i/D435if/D455/ D455f
Z16	Y8			Gyro & Accelerometer
Z16		UYVY		Gyro & Accelerometer
Z16	Y8		YUY2	Gyro & Accelerometer
Z16			YUY2	Gyro & Accelerometer

### A4. VL53L0X TOF sensor specifications

Target reflectance level (full FOV)	Conditions	Indoor (2)	Outdoor overcast (2)
White target (88%)	Typical	200cm+ (1)	80 cm
	Minimum	120 cm	60 cm
Grey target (17%)	Typical	80 cm	50 cm
	Minimum	70 cm	40 cm

Range profile	Range timing budget	Typical performance	Typical application
Default mode	30 ms	1.2 m, accuracy as per <a href="#">Table 12</a>	Standard
High accuracy	200 ms	1.2 m, accuracy < +/- 3%	Precise measurement
Long range	33 ms	2 m, accuracy as per <a href="#">Table 12</a>	Long ranging, only for dark conditions (no IR)
High speed	20 ms	1.2 m, accuracy +/- 5%	High speed where accuracy is not priority

## A5. Linear actuators specifications



<b>Stroke length</b>	10"
<b>Voltage</b>	12 V – 24 V
<b>Force</b>	Delivers 330 lbs of force
<b>Speed</b>	0.315 in/s at 12v; 0.670 in/s at 24v
<b>Extras</b>	Position feedback, Mounting brackets, Weather resistant, Limit switches
<b>Limit Switches</b>	Included - Automatically stops at end of travel
<b>Environment</b>	IP54 rated (Weather resistant)
<b>Feedback sensor</b>	Hall Effect - Each pulse represents 0.007046 inches (0.17896mm)
<b>RED wire</b>	Motor (+)
<b>BLACK wire</b>	Motor (-)
<b>GREEN wire</b>	Hall Sensor GND
<b>WHITE wire</b>	Hall Sensor POWER
<b>YELLOW wire</b>	Hall Sensor OUTPUT