Analysis, Fabrication and Experimentation of Delta robot

Summer Internship Report

Bachlor of Technology

in

Automation and Robotics

by

SHRAVAN A P: CB.EN.U4ARE22045
SACHIN S: CB.EN.U4ARE22048
Department of Mechanical Enginnering
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham, India

Under the guidance of

Dr. Anirban Nag

SCHOOL OF MECHATRONICS AND ROBOTICS
INDIAN INSTITUTE OF ENGINEERING SCIENCE AND TECHNOLOGY, SHIBPUR - 711103

MAY-JUNE 2025

ACKNOWLEDGEMENTS

First and foremost, we would like to express our heartfelt gratitude to our internship supervisor, **Dr. Anirban Nag**, Assistant Professor, School of Mechatronics and Robotics, IIEST Shibpur, for his exceptional mentorship, constant encouragement, and insightful guidance throughout the duration of our work. His expertise, critical feedback, and patient supervision have played a significant role in shaping the technical depth and direction of this project.

We are also sincerely thankful to **Prof. Dr. Tanmay Pal**, Professor and Head, School of Mechatronics and Robotics, IIEST Shibpur, for providing us with an academically rich and supportive environment that greatly facilitated our learning and exploration.

Our sincere appreciation also extends to all the faculty members and technical staff at IIEST Shibpur for their welcoming attitude, valuable input, and the access to lab resources that supported our progress during the internship.

We are especially grateful to the **Department of Mechanical Engineering, Amrita School of Engineering**, and to its **Head of Department**, for providing us the opportunity to undertake this internship and for their continuous encouragement and institutional support throughout.

We would also like to thank our faculty mentors and coordinators at Amrita School of Engineering for guiding us and helping coordinate this internship, which has been a significant learning milestone in our academic journey.

Lastly, we extend our deepest gratitude to our parents and families, whose unwavering support, trust, and encouragement have always inspired and empowered us to pursue excellence.

ABSTRACT

This report presents a complete study on the modeling, simulation, and implementation of two fundamental robotic mechanisms: the planar four-bar linkage and the Delta parallel robot. The work combines symbolic computation, dynamic simulation, and hardware prototyping to investigate and realize the motion behavior and control characteristics of both systems.

For the four-bar mechanism, inverse kinematics was derived to trace a predefined circular trajectory of the coupler point. Smooth path generation was achieved through cubic polynomial interpolation of angular variables, followed by dynamic modeling using constrained Euler-Lagrange formulations. Forward dynamics simulations were conducted to validate constraint satisfaction and system behavior, while inverse dynamics enabled computation of joint torque profiles.

In the case of the Delta robot, actuator-space dynamics were formulated using Jacobian-based constraint equations. The resulting symbolic model captured the full dynamic behavior of the platform and facilitated inverse dynamic torque estimation. The mechanical structure was modeled in Fusion 360 and fabricated using additive manufacturing with PLA material. Actuation was implemented using three Dynamixel AX-12A servos controlled via Arduino Uno boards, and communication delays and synchronization issues were addressed during testing.

This integrated workflow — from mathematical modeling to real-time implementation — bridges the gap between theoretical robotics and hands-on engineering. The report offers a validated framework for constrained multibody dynamics and real-world robotic design, laying a solid foundation for future extensions in control, trajectory optimization, or high-speed pick-and-place applications.

Keywords: DELTA robot, 4-Bar mechanism, kinematics, dynamics, parallel manipulator, Wolfram Mathematica, robot design, CAD, fabrication, workspace analysis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The field of robotics is a dynamic convergence of mechanical design, control theory, and computational intelligence. Within this domain, two foundational systems — the **Delta parallel manipulator** and the **four-bar linkage mechanism** — hold significant academic and industrial relevance. The present report undertakes a detailed study of these two mechanisms from both kinematic and dynamic perspectives, blending theoretical modelling with simulation and real-world prototyping.

The **Delta robot** [2], originally developed to meet high-speed automation requirements, exemplifies the advantages of parallel architectures: high stiffness-to-weight ratios, precise end-effector positioning, and superior dynamic performance. It is extensively used in industries such as packaging, electronics, and pharmaceuticals where rapid and accurate pick-and-place operations are crucial. Its symmetric RUU kinematic structure ensures pure translational motion of the end-effector, which is both mechanically efficient and computationally tractable.

In contrast, the **four-bar linkage mechanism** is the most fundamental planar closed-chain linkage in mechanism design. It is widely employed in various applications, including engine systems, robotics arms, and biomechanical devices, owing to its simplicity and versatility. Despite its simplicity, its analysis—particularly when extended to dynamic conditions—offers deep insight into system behavior, control effort, and actuation requirements.

This report systematically explores the geometry, kinematics, and dynamics of both mechanisms using a rigorous analytical approach supported by symbolic computation. The modeling and simulation were carried out primarily using **Wolfram Mathematica**, which enabled not only the symbolic derivation of motion and force equations but also efficient numerical simulation of both forward and inverse dynamics.

Objectives of the Study:

- To derive and solve the inverse and forward kinematics of a 3-DOF Delta parallel robot.
- To formulate and simulate the forward and inverse dynamics using constrained Euler-Lagrange methods.
- To analyze the workspace and trajectory planning of both mechanisms with singularity considerations.
- To design and fabricate a prototype Delta robot using CAD and additive manufacturing techniques.
- To validate the analytical and simulation results with visualizations and actuator control experiments.

Methodology Overview: The investigation follows a sequential flow starting with the derivation of constraint equations and Jacobians, followed by Lagrangian formulation for dynamics. Symbolic tools are used for algebraic manipulation and matrix computation. The resulting motion and force profiles are validated through dynamic simulations. For real-world correlation, the Delta robot was modeled in `Fusion 360`, 3D-printed, and controlled via Arduino-based actuation.

The integration of computation, prototyping, and simulation allows this report to serve not only as a theoretical analysis but also as a practical guide for constructing parallel manipulators. Ultimately, this project aims to bridge the gap between abstract modeling and hands-on implementation—an essential stride in robotics research and development.

Chapter 2

Delta Robot Manipulator Geometry and Inverse Kinematics

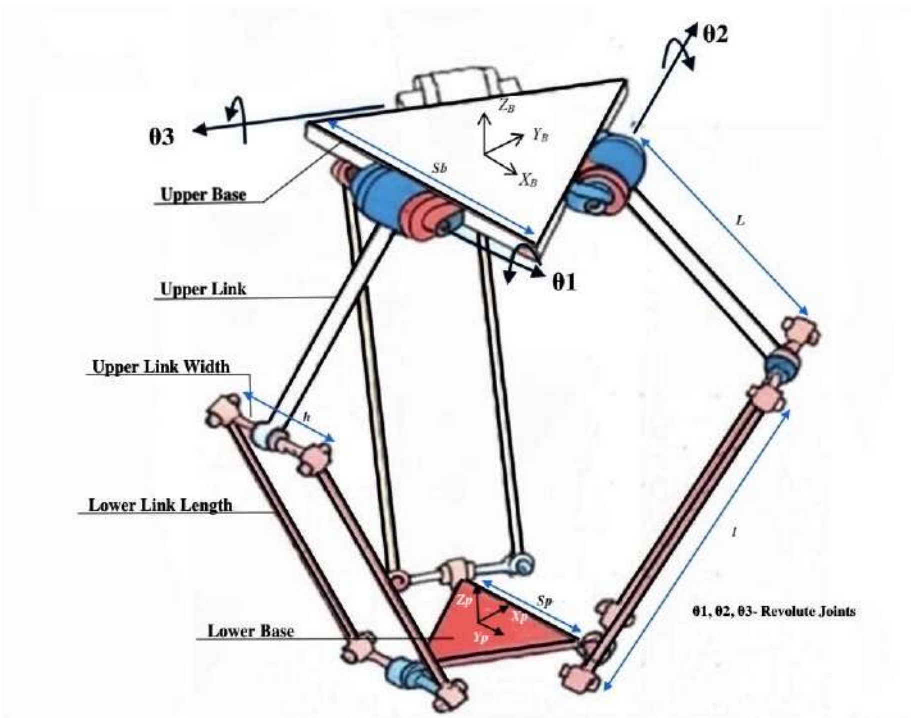


Figure 2.1: Geometry of the delta robot manipulator adapted from [1]

2.1 Geometry and Kinematic Architecture

The delta robot represents a sophisticated parallel manipulator characterized by its unique three-dimensional kinematic architecture. This mechanism consists of three identical RUU (Revolute-Universal-Universal) kinematic chains that connect a fixed triangular base platform to a mobile end-effector platform. Each kinematic chain comprises an actuated revolute joint at the base, followed by two universal joints that provide the necessary degrees of freedom for spatial motion.

The geometric configuration of the delta robot is fundamentally based on equilateral triangular arrangements for both the base and end-effector platforms. This symmetric design ensures uniform workspace characteristics and balanced dynamic properties across all three kinematic chains. The base platform is positioned at the upper level with three actuated revolute joints, while the end-effector platform is located at the lower level, connected through three parallel kinematic chains.

Each kinematic chain consists of two primary links: the proximal link (upper arm) directly connected to the actuated joint at the base, and the distal link (lower arm) connected to the end-effector platform through universal joints. The proximal links are constrained to move in vertical planes passing through the center of the base platform, while the distal links maintain parallel orientation to the base platform throughout the motion.

2.1.1 Geometric Parameters and Coordinate Systems

The delta robot's geometric parameters are precisely defined to establish the kinematic relationships between the base platform, kinematic chains, and end-effector platform. The coordinate system is established with the origin at the center of the base platform, with the z-axis pointing vertically downward toward the workspace.

Table 2.1: Delta Robot Geometric Parameters

Component	**Parameter**	**Symbol**	**Description**
Upper Platform	Circumradius	w_b	Distance from center to base joint
End-Effector Platform	Circumradius	u_p	Distance from center to platform joint
Proximal Link	Length	l	Upper arm link length
Distal Link	Length	L	Lower arm link length
Base Platform	Side spacing	S_p	Base platform geometry parameter
End-Effector	Platform width	W_p	End-effector platform parameter
Joint Angles	Active angles	$\theta_1, \theta_2, \theta_3$	Actuated revolute joint angles

The base platform vertices are positioned at angles of 0, 120, and 240 from the positive x-axis, creating a symmetric triangular configuration. Similarly, the end-effector platform maintains this triangular geometry but with a smaller circumradius to optimize the workspace characteristics.

2.2 Degree of Freedom Analysis

The mobility analysis of the delta robot is conducted using the Grübler-Kutzbach criterion for spatial mechanisms. This analysis is essential for understanding the kinematic constraints and determining the number of independent actuators required for complete control of the end-effector motion.

For a spatial mechanism, the degrees of freedom are calculated using:

$$\text{DOF} = 6(N-1) - \sum_{i=1}^{j}(6 - f_i) \tag{2.1}$$

where:

- N = total number of links (including the fixed base)
- j = total number of joints
- f_i = degrees of freedom of joint i

For the delta robot configuration:

- Moving links: 6 (two per kinematic chain) + 1 (end-effector platform) = 7 links
- Total links including base: $N = 7 + 1 = 8$ links
- Revolute joints at base: 3 joints with 1 DOF each
- Universal joints: 6 joints with 2 DOF each (two per kinematic chain)

Substituting into the mobility equation:

$$\begin{aligned}
\text{DOF} &= 6(8-1) - [3 \times (6-1) + 6 \times (6-2)] && (2.2)\\
&= 6 \times 7 - [3 \times 5 + 6 \times 4] && (2.3)\\
&= 42 - [15 + 24] && (2.4)\\
&= 42 - 39 && (2.5)\\
&= 3 && (2.6)
\end{aligned}$$

Thus, the delta robot possesses three degrees of freedom, corresponding to:

- Translation along the x-axis: x
- Translation along the y-axis: y
- Translation along the z-axis: z

This three-degree-of-freedom configuration makes the delta robot ideal for pick-and-place operations, high-speed assembly tasks, and precision positioning applications where pure translational motion is required.

2.3 Inverse Kinematics Formulation

The inverse kinematics problem for the delta robot involves determining the three actuated joint angles θ_1, θ_2, and θ_3 given the desired position (x, y, z) of the end-effector platform. This process requires solving a system of nonlinear equations derived from the geometric constraints of each kinematic chain.

2.3.1 Kinematic Constraint Equations

The kinematic analysis begins by establishing the position vectors for each component of the delta robot. The base joint positions are defined as:

$$\mathbf{b}_1 = \{0, -w_b, 0\} \tag{2.7}$$

$$\mathbf{b}_2 = \mathbf{R}_z \cdot \mathbf{b}_1 = \left\{\frac{\sqrt{3}w_b}{2}, \frac{w_b}{2}, 0\right\} \tag{2.8}$$

$$\mathbf{b}_3 = \mathbf{R}_z^2 \cdot \mathbf{b}_1 = \left\{-\frac{\sqrt{3}w_b}{2}, \frac{w_b}{2}, 0\right\} \tag{2.9}$$

where $\mathbf{R}_z$ represents a rotation matrix about the z-axis by $\frac{2\pi}{3}$ radians.
The end-effector platform joint positions are defined as:

$$\mathbf{p}_1 = \{0, -u_p, 0\} + \{x, y, z\} \tag{2.10}$$

$$\mathbf{p}_2 = \left\{\frac{S_p}{2}, W_p, 0\right\} + \{x, y, z\} \tag{2.11}$$

$$\mathbf{p}_3 = \left\{-\frac{S_p}{2}, \frac{W_p}{2}, 0\right\} + \{x, y, z\} \tag{2.12}$$

For each kinematic chain, the proximal link endpoint positions are calculated using rotation matrices:

$$\mathbf{L}_1 = \mathbf{R}_{x2} \cdot \mathbf{R}_{x1}(\theta_1) \cdot \{0, 0, l\} = \{0, -l\cos\theta_1, -l\sin\theta_1\} \tag{2.13}$$

$$\mathbf{L}_2 = \mathbf{R}_z \cdot \mathbf{R}_{x2} \cdot \mathbf{R}_{x1}(\theta_2) \cdot \{0, 0, l\} \tag{2.14}$$

$$\mathbf{L}_3 = \mathbf{R}_z^2 \cdot \mathbf{R}_{x2} \cdot \mathbf{R}_{x1}(\theta_3) \cdot \{0, 0, l\} \tag{2.15}$$

The constraint that each distal link must have length L leads to the fundamental kinematic equations:

$$\begin{aligned} f_1 = {} & l^2 + u_p^2 - 2u_p w_b + w_b^2 + x^2 - 2u_p y + 2w_b y + y^2 + z^2 \\ & - 2l(u_p - w_b - y)\cos\theta_1 + 2lz\sin\theta_1 = L^2 \end{aligned} \tag{2.16}$$

$$
\begin{aligned}
f_2 = l^2 + \frac{S_p^2}{4} - \frac{\sqrt{3}}{2}S_p w_b + w_b^2 - w_b W_p + W_p^2 + S_p x - \sqrt{3}w_b x + x^2 \\
- w_b y + 2W_p y + y^2 + z^2 - \frac{l}{2}\left(\sqrt{3}S_p + 2(-2w_b + W_p + \sqrt{3}x + y)\right)\cos\theta_2 \\
+ 2lz\sin\theta_2 = L^2
\end{aligned} \tag{2.17}
$$

$$
\begin{aligned}
f_3 = l^2 + \frac{S_p^2}{4} - \frac{\sqrt{3}}{2}S_p w_b + w_b^2 - \frac{w_b W_p}{2} + \frac{W_p^2}{4} - S_p x + \sqrt{3}w_b x + x^2 \\
- w_b y + W_p y + y^2 + z^2 - \frac{l}{2}\left(\sqrt{3}S_p - 4w_b + W_p - 2\sqrt{3}x + 2y\right)\cos\theta_3 \\
+ 2lz\sin\theta_3 = L^2
\end{aligned} \tag{2.18}
$$

2.3.2 Analytical Solution Method

Each constraint equation can be rearranged into the standard form:

$$A_i \cos\theta_i + B_i \sin\theta_i + C_i = 0 \quad \text{for } i = 1, 2, 3 \tag{2.19}$$

Using the half-angle substitution $t_i = \tan(\theta_i/2)$, the trigonometric functions become:

$$\cos\theta_i = \frac{1 - t_i^2}{1 + t_i^2}, \quad \sin\theta_i = \frac{2t_i}{1 + t_i^2} \tag{2.20}$$

Substituting these expressions and clearing denominators yields quadratic equations in t_i:

$$(C_i - A_i)t_i^2 + 2B_i t_i + (A_i + C_i) = 0 \tag{2.21}$$

The solutions are obtained using the quadratic formula:

$$t_i = \frac{-2B_i \pm \sqrt{4B_i^2 - 4(C_i - A_i)(A_i + C_i)}}{2(C_i - A_i)} \tag{2.22}$$

Finally, the joint angles are recovered using:

$$\theta_i = 2\arctan(t_i) \tag{2.23}$$

This analytical approach provides up to two solutions for each joint angle, corresponding to different assembly configurations of the delta robot.

2.4 Numerical Implementation and Validation

The inverse kinematics solution is implemented numerically using the geometric parameters specified in the Mathematica analysis. The parameter values used for validation are:

$$\begin{aligned}\text{Data} = [&S_p \to 76 \times 10^{-3}, \quad l \to 524 \times 10^{-3}, \quad L \to 1244 \times 10^{-3}, \\ &w_b \to 164 \times 10^{-3}, \quad W_p \to 22 \times 10^{-3}, \quad u_p \to 44 \times 10^{-3}]\end{aligned} \tag{2.24}$$

2.4.1 Solution Example

For the target end-effector position $(x, y, z) = (0, 0, -0.9)$ meters, the inverse kinematics solution yields:

First Kinematic Chain:

$$t_1 = \{-0.181105, -3.10347\} \tag{2.25}$$

$$\theta_1 = \{-0.358327, -2.51816\} \text{ radians} \tag{2.26}$$

Second Kinematic Chain:

$$t_2 = \{-0.181037, -3.10312\} \tag{2.27}$$

$$\theta_2 = \{-0.358194, -2.5181\} \text{ radians} \tag{2.28}$$

Third Kinematic Chain:

$$t_3 = \{-0.176831, -3.0828\} \tag{2.29}$$

$$\theta_3 = \{-0.350043, -2.51425\} \text{ radians} \tag{2.30}$$

2.4.2 Configuration Analysis

The multiple solutions obtained for each joint angle correspond to different assembly modes of the delta robot. These configurations represent the "elbow-up" and "elbow-down" positions for each kinematic chain, providing flexibility in choosing the optimal configuration based on workspace constraints, singularity avoidance, and dynamic performance requirements.

The existence of multiple solutions is characteristic of parallel manipulators and reflects the complex kinematic relationships inherent in these mechanisms. In practical applications, the selection of the appropriate configuration depends on factors such as:

- Workspace accessibility and obstacle avoidance
- Proximity to kinematic singularities

- Joint limit constraints
- Dynamic performance optimization
- Trajectory continuity requirements

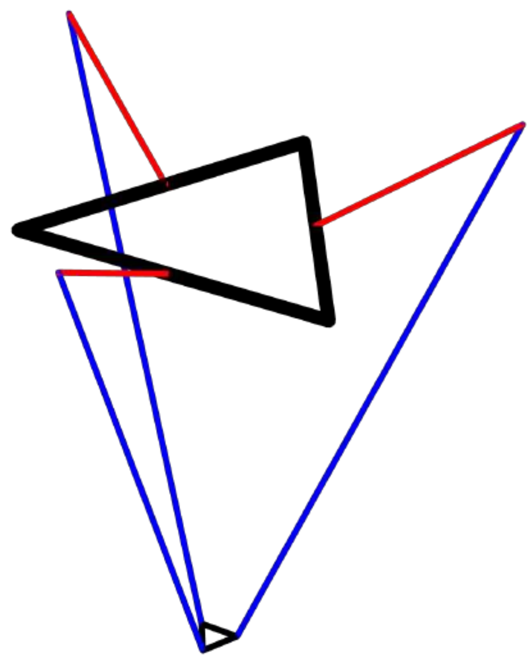


Figure 2.2: Geometric plot of delta robot

Chapter 3

CAD Modeling and Workspace Analysis

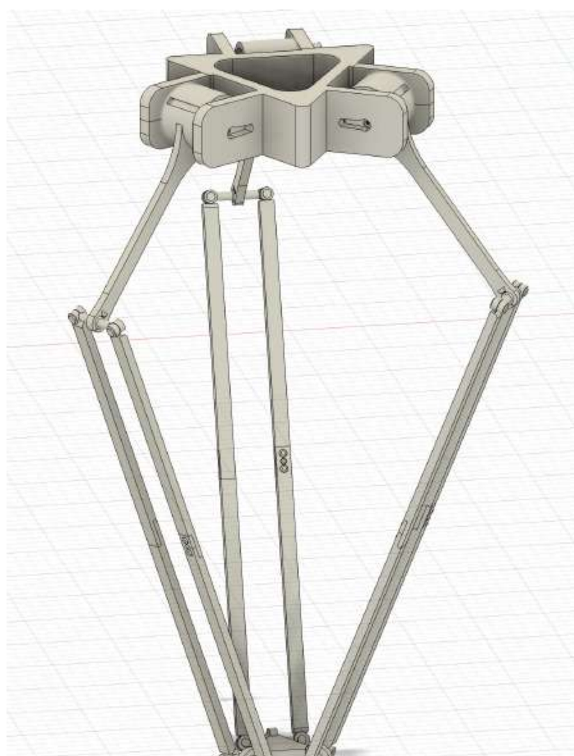


Figure 3.1: 3-DOF Delta Robot CAD Model in Fusion 360

This chapter describes the complete CAD modeling process of the Delta parallel robot and subsequent workspace analysis. The model follows an RUU (Revolute-Universal-Universal) kinematic chain architecture.

3.1 CAD Modeling

3.1.1 Software

Fusion 360 was used for modeling, assembling, and simulating the manipulator. Key features utilized include:

- Parametric 3D modeling
- Assembly design with joint constraints
- Motion simulation
- Manufacturing drawing generation

3.1.2 Dimensional Analysis

Critical dimensions extracted from manufacturing drawings:

Table 3.1: Delta Robot Component Dimensions

Component	**Parameter**	**Value (mm)**
Upper Base	Width	581.97
	Height	504.00
Upper Link	Length	524.00
	Joint Spacing	76.00
Lower Link	Length	327.00
	Width	44.00
End Effector	Platform Radius	164.00
	Joint Offset	22.00

3.1.3 Modeling Workflow

The modeling process followed these stages:

1. **Part Creation**:
   - Base plate with mounting holes
   - Proximal and distal links with joint features
   - End effector platform
2. **Assembly**:
   - Grounded base component

- Revolute joints at base (θ_1, θ_2, θ_3)
- Universal joints at intermediate connections
- Spherical joints at end effector

3. **Validation**:

- Range of motion simulation
- Interference checking
- Joint limit verification

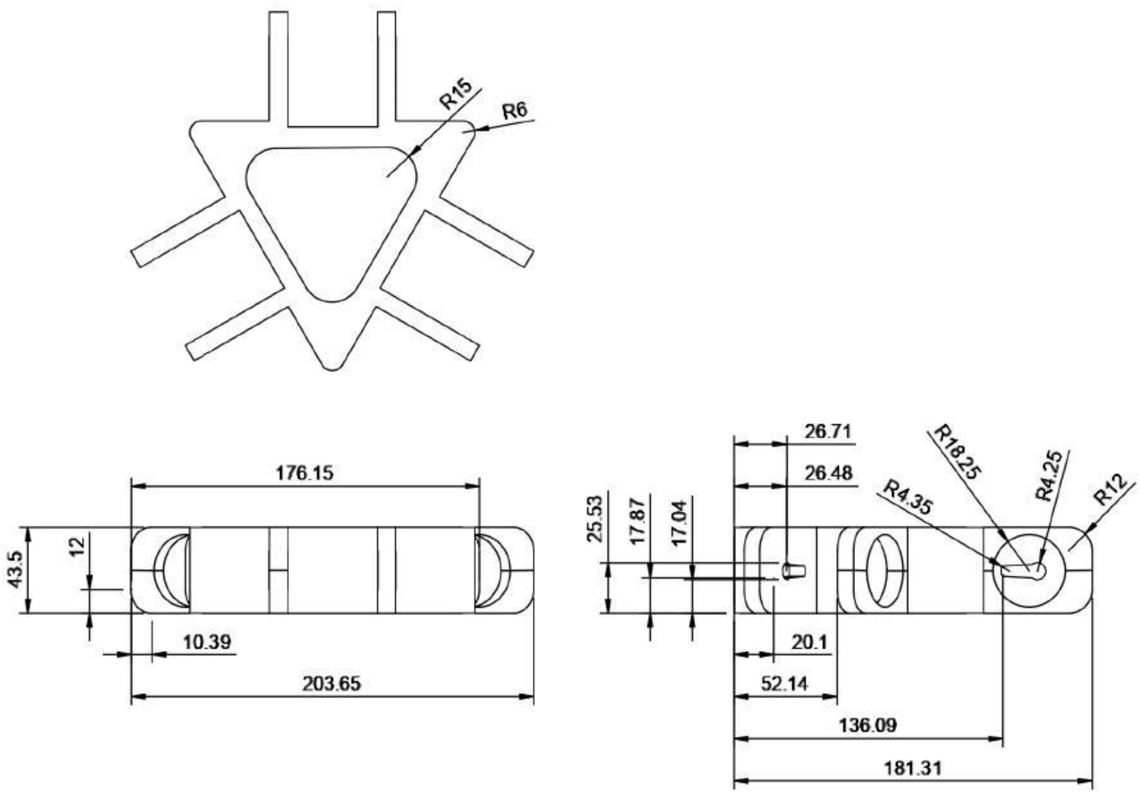


Figure 3.2: Upper Base Component

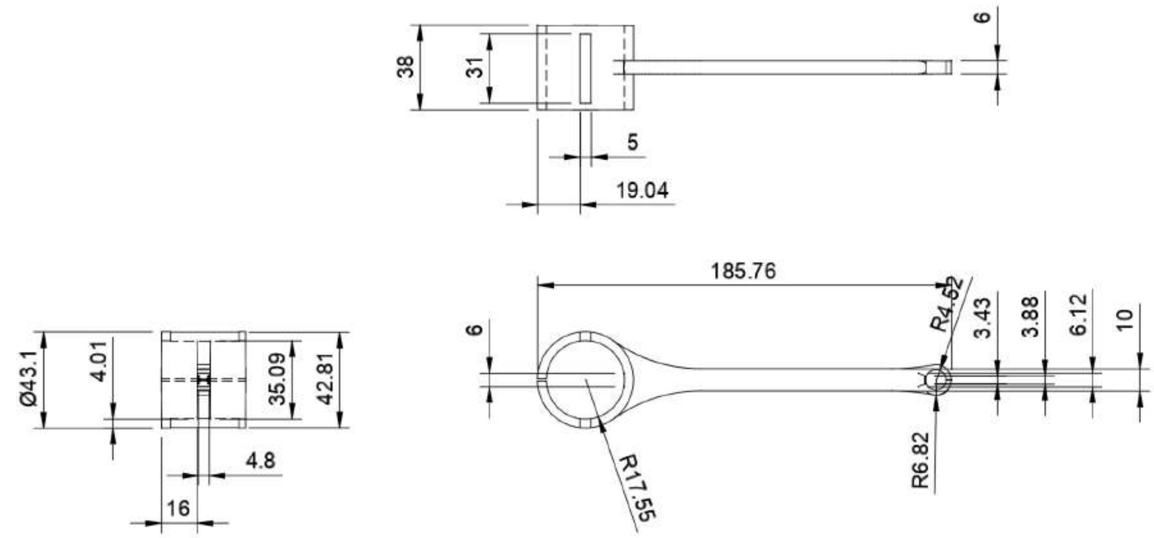


Figure 3.3: Upper Link Assembly

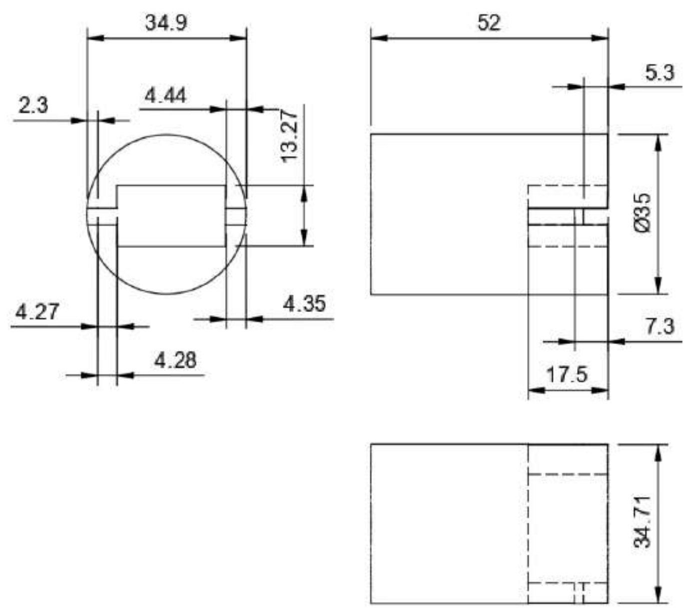


Figure 3.4: Joint Shaft Detail

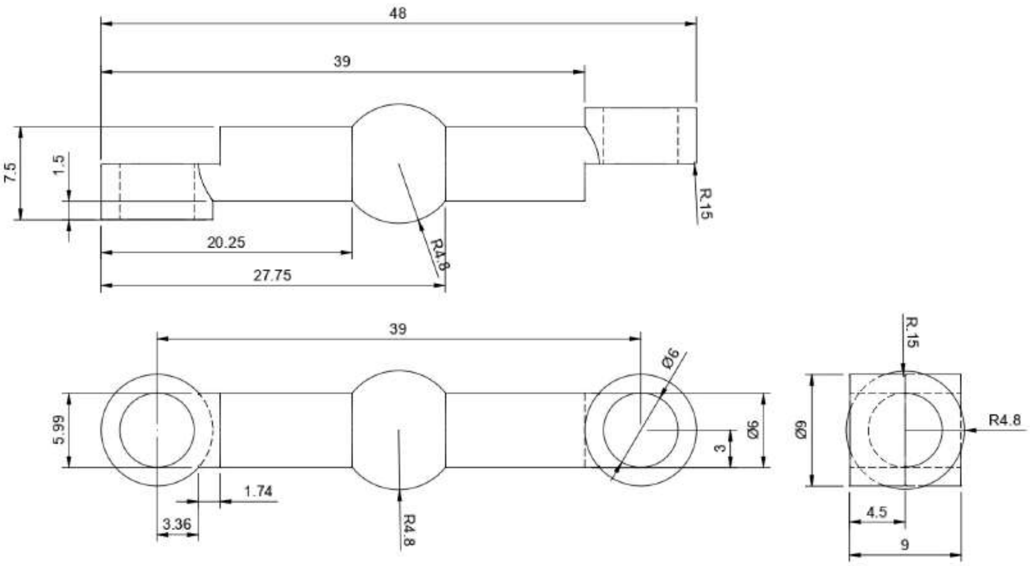


Figure 3.5: Lower Link Width Dimension

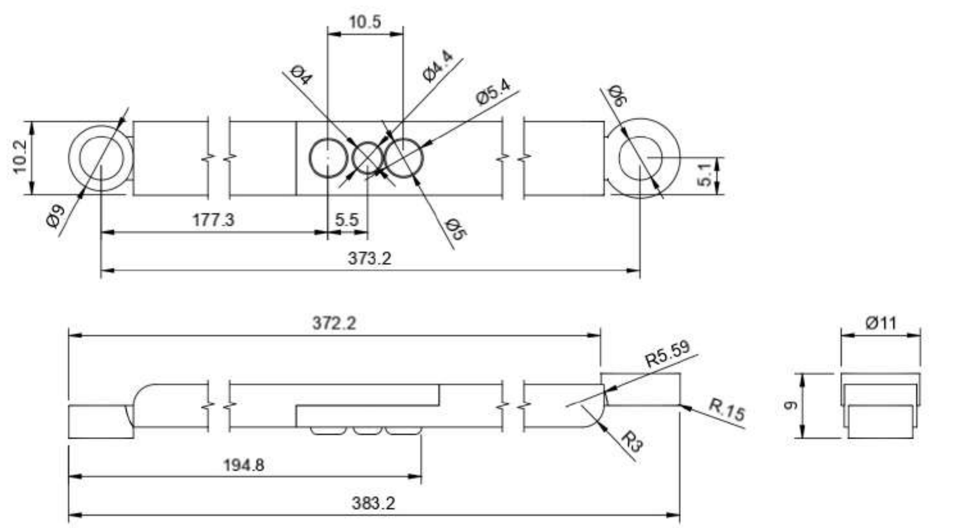


Figure 3.6: Lower Link Assembly

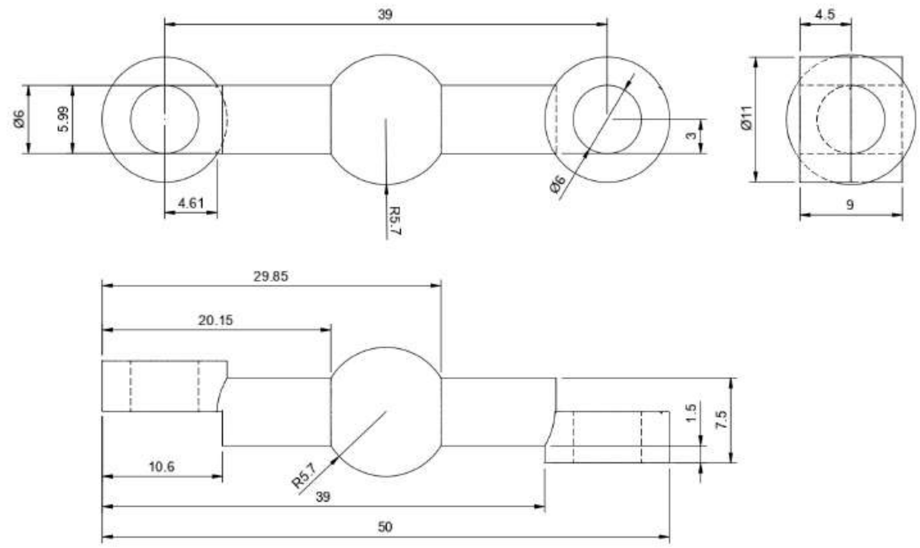


Figure 3.7: Lower Base Width

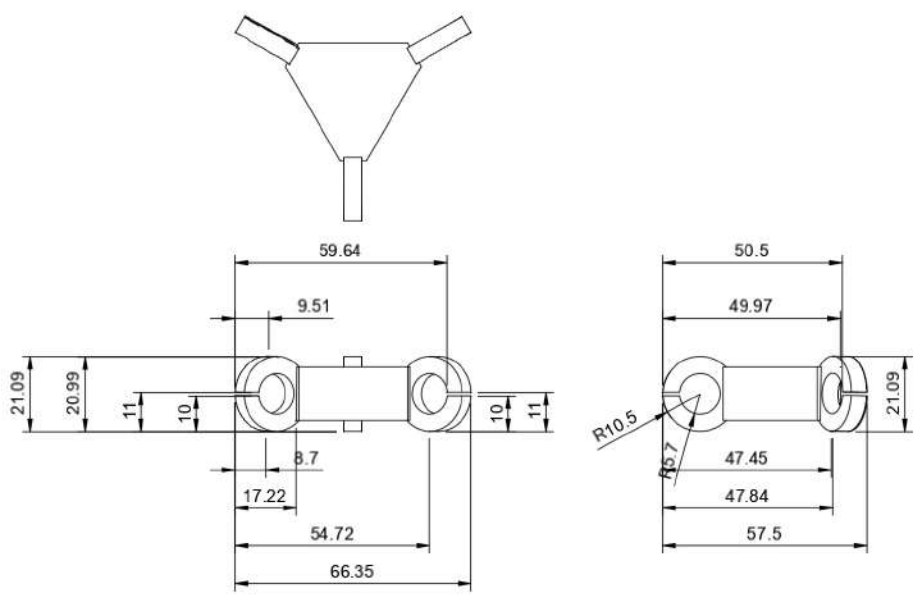


Figure 3.8: Lower Base Assembly

3.2 Workspace Analysis

The workspace boundary equations were derived for all three kinematic chains of the RUU parallel mechanism. Due to the 120° rotational symmetry, the equations for chains 2 and 3 are rotational transformations of the chain 1 solution.

3.2.1 General Formulation

For each kinematic chain i ($i = 1, 2, 3$), the position equation is:

$$\|\mathbf{b}_i + l\mathbf{u}_i - (\mathbf{p}_i + \mathbf{P})\| = L \tag{3.1}$$

Where:

- $\mathbf{P} = [x, y, z]^T$ = End-effector position
- $\mathbf{b}_i$ = Base joint positions
- $\mathbf{p}_i$ = End-effector attachment points
- $\mathbf{u}_i = [0, -\cos\theta_i, -\sin\theta_i]^T$ = Proximal link direction
- l = Proximal link length (524 mm)
- L = Distal link length (1244 mm)

The base and end-effector coordinates are:

$$\mathbf{b}_1 = [0, -\mathrm{wb}, 0]^T \tag{3.2}$$
$$\mathbf{b}_2 = \mathbf{R}_z(120°)\mathbf{b}_1 \tag{3.3}$$
$$\mathbf{b}_3 = \mathbf{R}_z(240°)\mathbf{b}_1 \tag{3.4}$$
$$\mathbf{p}_1 = [0, -\mathrm{up}, 0]^T \tag{3.5}$$
$$\mathbf{p}_2 = [\mathrm{Sp}/2, \mathrm{Wp}, 0]^T \tag{3.6}$$
$$\mathbf{p}_3 = [-\mathrm{Sp}/2, \mathrm{Wp}/2, 0]^T \tag{3.7}$$

With rotation matrix $\mathbf{R}_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

3.2.2 Chain-Specific Boundary Equations

3.2.2.1 Chain 1 Boundary Equation

After tangent half-angle substitution and discriminant condition:

$$16l^2z^2 = 4\begin{bmatrix} l^2 - L^2 + 2l\mathrm{up} + \mathrm{up}^2 - 2l\mathrm{wb} \\ -2\mathrm{upwb} + \mathrm{wb}^2 + x^2 - 2ly \\ -2\mathrm{up}y + 2\mathrm{wb}y + y^2 + z^2 \end{bmatrix} \times \begin{bmatrix} l^2 - L^2 - 2l\mathrm{up} + \mathrm{up}^2 + 2l\mathrm{wb} \\ -2\mathrm{upwb} + \mathrm{wb}^2 + x^2 + 2ly \\ -2\mathrm{up}y + 2\mathrm{wb}y + y^2 + z^2 \end{bmatrix} \tag{3.8}$$

With parameter substitution (mm $\to$ m):

$$\frac{68644z^2}{15625} = 4\left(-\tfrac{4326}{3125} + x^2 - \tfrac{101y}{125} + y^2 + z^2\right)\left(-\tfrac{708}{625} + x^2 + \tfrac{161y}{125} + y^2 + z^2\right) \tag{3.9}$$

3.2.2.2 Chain 2 Boundary Equation

For the 120°-rotated chain:

$$16l^2z^2 = 4\begin{bmatrix} l^2 - L^2 + l\sqrt{3}\mathrm{Sp} + \frac{\mathrm{Sp}^2}{4} + 2l\mathrm{wb} \\ -l\mathrm{Wp} - \sqrt{3}\mathrm{Spwb} + \mathrm{wb}^2 - \mathrm{wbWp} \\ +\mathrm{Wp}^2 + \frac{1}{4}(\sqrt{3}\mathrm{Sp} + 2x - 2\sqrt{3}y)^2 \\ +\frac{1}{4}(-\mathrm{Sp} + 2\sqrt{3}x + 2y)^2 + z^2 \end{bmatrix} \times \begin{bmatrix} l^2 - L^2 - l\sqrt{3}\mathrm{Sp} + \frac{\mathrm{Sp}^2}{4} - 2l\mathrm{wb} \\ +l\mathrm{Wp} - \sqrt{3}\mathrm{Spwb} + \mathrm{wb}^2 - \mathrm{wbWp} \\ +\mathrm{Wp}^2 + \frac{1}{4}(-\sqrt{3}\mathrm{Sp} + 2x + 2\sqrt{3}y)^2 \\ +\frac{1}{4}(\mathrm{Sp} + 2\sqrt{3}x - 2y)^2 + z^2 \end{bmatrix} \tag{3.10}$$

With parameter substitution:

$$\frac{1998304z^2}{15625} = 4\begin{pmatrix} \frac{5437}{1250} - \frac{1634\sqrt{3}}{15625} - \frac{38x}{125} \\ +\frac{344\sqrt{3}x}{125} - \frac{322y}{125} + 4x^2 + 4y^2 + 4z^2 \end{pmatrix}\begin{pmatrix} \frac{176011}{31250} + \frac{171\sqrt{3}}{3125} - \frac{38x}{125} \\ -\frac{36\sqrt{3}x}{125} + \frac{202y}{125} + 4x^2 + 4y^2 + 4z^2 \end{pmatrix} \tag{3.11}$$

3.2.2.3 Chain 3 Boundary Equation

For the 240°-rotated chain:

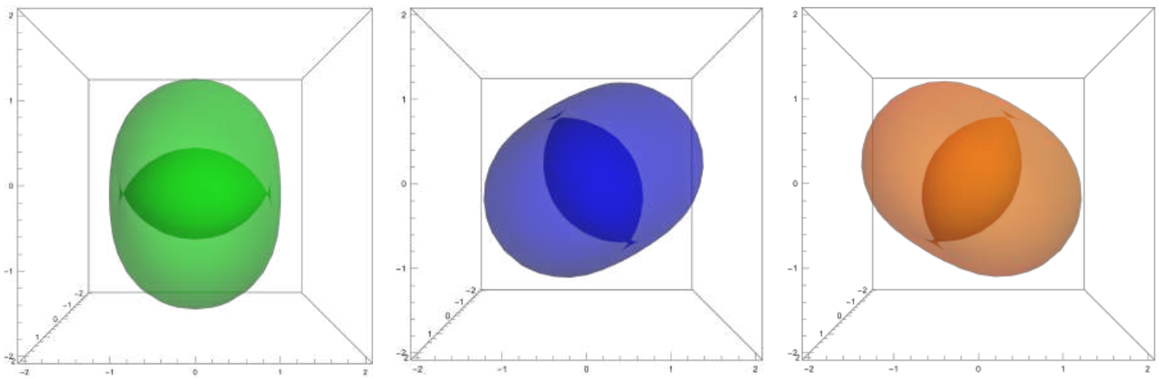
$$16l^2z^2 = 4\begin{bmatrix} l^2 - L^2 - l\sqrt{3}\mathrm{Sp} + \frac{\mathrm{Sp}^2}{4} + 2l\mathrm{wb} \\ -\frac{l\mathrm{Wp}}{2} + \sqrt{3}\mathrm{Spwb} + \mathrm{wb}^2 - \frac{\mathrm{wbWp}}{2} \\ +\frac{\mathrm{Wp}^2}{4} + \frac{1}{4}(-\sqrt{3}\mathrm{Sp} - 2x + 2\sqrt{3}y)^2 \\ +\frac{1}{4}(\mathrm{Sp} - 2\sqrt{3}x - 2y)^2 + z^2 \end{bmatrix} \times \begin{bmatrix} l^2 - L^2 + l\sqrt{3}\mathrm{Sp} + \frac{\mathrm{Sp}^2}{4} - 2l\mathrm{wb} \\ +\frac{l\mathrm{Wp}}{2} + \sqrt{3}\mathrm{Spwb} + \mathrm{wb}^2 - \frac{\mathrm{wbWp}}{2} \\ +\frac{\mathrm{Wp}^2}{4} + \frac{1}{4}(\sqrt{3}\mathrm{Sp} - 2x - 2\sqrt{3}y)^2 \\ +\frac{1}{4}(-\mathrm{Sp} - 2\sqrt{3}x + 2y)^2 + z^2 \end{bmatrix} \tag{3.12}$$

With parameter substitution:

$$\frac{1098304z^2}{15625} = 4\begin{pmatrix} \frac{216039}{59000} - \frac{1634\sqrt{3}}{15625} - \frac{38x}{125} \\ +\frac{344\sqrt{3}x}{125} - \frac{333y}{125} + 4x^2 + 4y^2 + 4z^2 \end{pmatrix}\begin{pmatrix} \frac{1412411}{250000} + \frac{171\sqrt{3}}{3125} - \frac{38x}{125} \\ -\frac{36\sqrt{3}x}{125} + \frac{191y}{125} + 4x^2 + 4y^2 + 4z^2 \end{pmatrix} \tag{3.13}$$

3.2.3 Workspace Visualization

To understand the contribution of each kinematic chain to the overall reachable space, the workspace boundaries were first computed individually for each chain and then for their combinations. The visualization was performed using 3D contour plotting of the nonlinear boundary equations derived earlier.

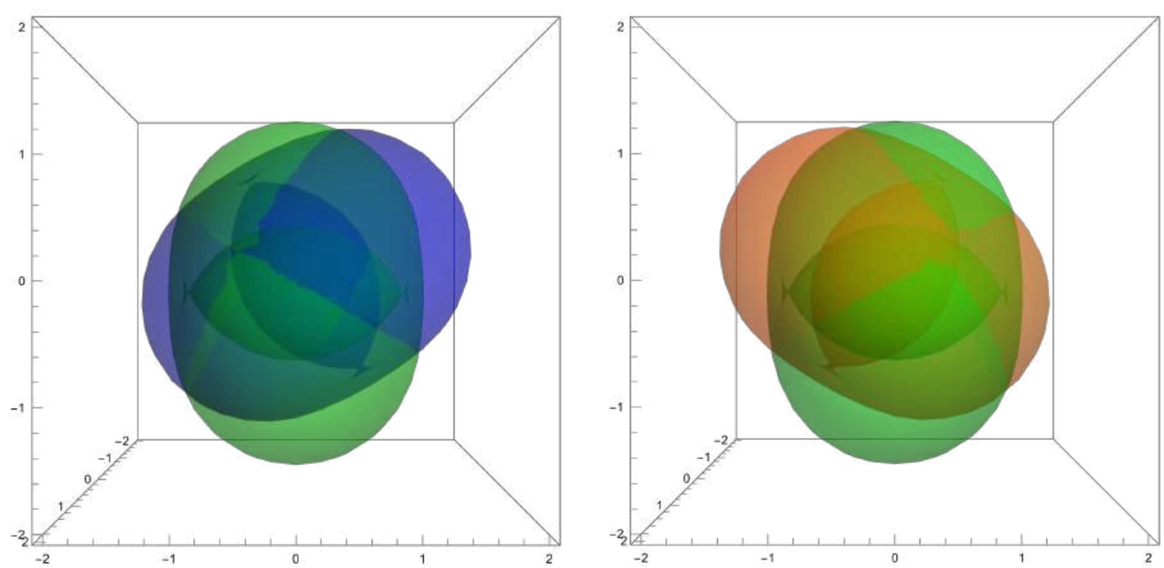


(a) Workspace of Chain 1 (R1)

(b) Workspace of Chain 2 (R2)

(c) Workspace of Chain 3 (R3)

Figure 3.9: Individual workspace boundaries for each chain



(d) Combined Workspace: R1 × R2

(e) Combined Workspace: R1 × R3

Figure 3.10: Pairwise workspace intersections to analyze overlapping reach zones

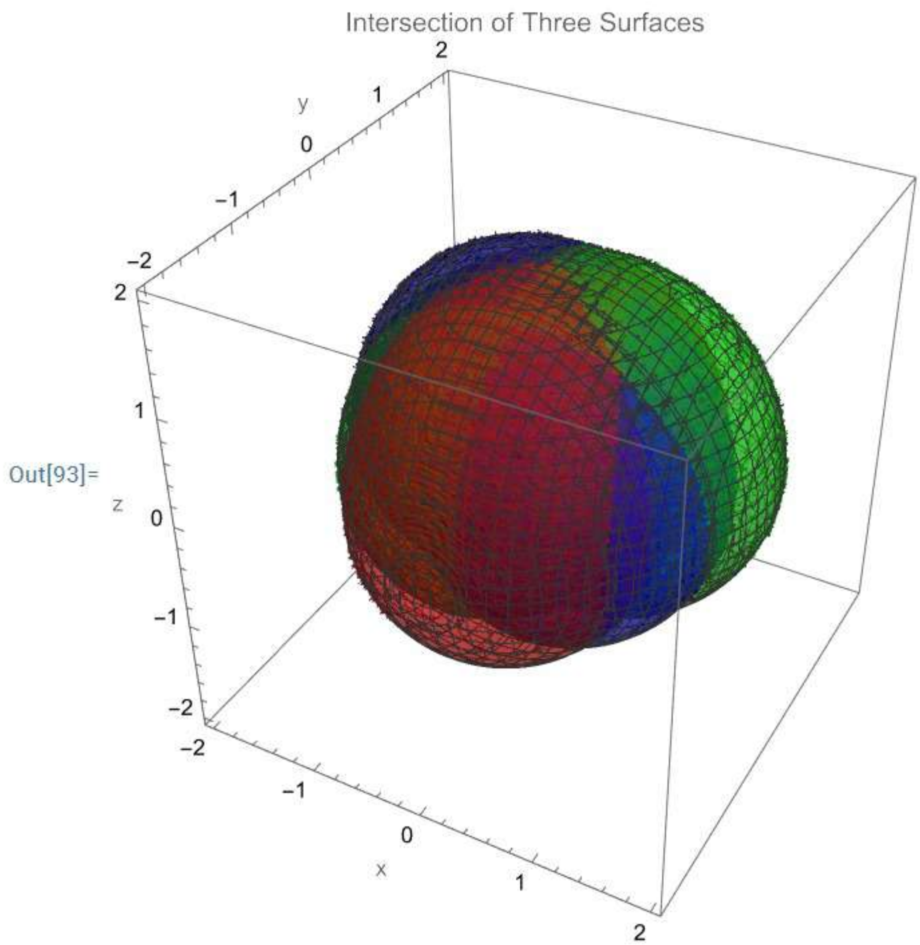


Figure 3.11: Complete Workspace Boundary Surface Visualization
(Green: R1, Blue: R2, Red: R3)

Mathematica implementation for pairwise intersections:

```
(* Chain 1-2 intersection *)
ContourPlot3D[{R1, R2}, {x, -0.5, 0.5}, {y, -0.5, 0.5}, {z,
   -1.2, -0.6},
  Mesh -> None, ContourStyle -> {Opacity[0.4, Green], Opacity
     [0.4, Blue]}]

(* Chain 1-3 intersection *)
ContourPlot3D[{R1, R3}, {x, -0.5, 0.5}, {y, -0.5, 0.5}, {z,
   -1.2, -0.6},
  Mesh -> None, ContourStyle -> {Opacity[0.4, Green], Opacity
     [0.4, Orange]}]
```

Chapter 4

Dynamics of planar and spatial closed-loop mechanisms

This chapter presents a comprehensive analysis of the trajectory tracking and dynamics simulation of a planar four-bar mechanism. The study is carried out through symbolic formulation of the kinematic and dynamic equations, followed by numerical simulation using forward dynamics. Additionally, actuator torque requirements are extracted using inverse dynamics based on the simulation response.

4.1 Loop closure and constraint Jacobian

The four-bar mechanism is a closed kinematic chain and is subject to position-level loop-closure constraints:

$$\eta_1 = -l_0 + l_1 \cos\theta_1 + l_2 \cos\theta_2 - l_3 \cos\theta_3 = 0, \tag{4.1}$$

$$\eta_2 = l_1 \sin\theta_1 + l_2 \sin\theta_2 - l_3 \sin\theta_3 = 0 \tag{4.2}$$

Differentiating the constraints with respect to time gives the Jacobian matrix:

$$\mathbf{J}_\eta = \begin{bmatrix} -l_1 \sin\theta_1 & -l_2 \sin\theta_2 & l_3 \sin\theta_3 \\ l_1 \cos\theta_1 & l_2 \cos\theta_2 & -l_3 \cos\theta_3 \end{bmatrix} \tag{4.3}$$

Each link is modeled as a rigid rod with:

$$I_i = \frac{1}{12} m_i l_i^2 \tag{4.4}$$

The kinetic energy is:

$$T = \frac{1}{6} m_1 l_1^2 \dot{\theta}_1^2 + \frac{1}{6} m_2 \left(3 l_1^2 \dot{\theta}_1^2 + 2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) + l_2^2 \dot{\theta}_2^2\right) + \frac{1}{6} m_3 l_3^2 \dot{\theta}_3^2 \tag{4.5}$$

The potential energy is:

$$V = \frac{1}{2} g l_1 m_1 \sin\theta_1 + g l_1 m_2 \sin\theta_1 + \frac{1}{2} g l_2 m_2 \sin\theta_2 + \frac{1}{2} g l_3 m_3 \sin\theta_3 \tag{4.6}$$

4.2 Euler-Lagrange equation of motion for a constrained mechanism

The Lagrangian $L = T - V$ leads to the constrained equations of motion:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = \sum_j \lambda_j J_{\eta,ji} \tag{4.7}$$

Let the generalized coordinate vector be:

$$q = [\theta_1, \theta_2, \theta_3]^T \tag{4.8}$$

Then, the equations of motion are expressed as:

$$\begin{bmatrix} \mathbf{M} & -\mathbf{J}_\eta^T \\ \mathbf{J}_\eta & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\mathbf{C}\dot{q} - \mathbf{G} \\ -\dot{\mathbf{J}}_\eta \dot{q} \end{bmatrix} \tag{4.9}$$

The simulation is carried out using Mathematica's `NDSolve` to integrate the system over $t \in [0, 5]$ s with known initial conditions.

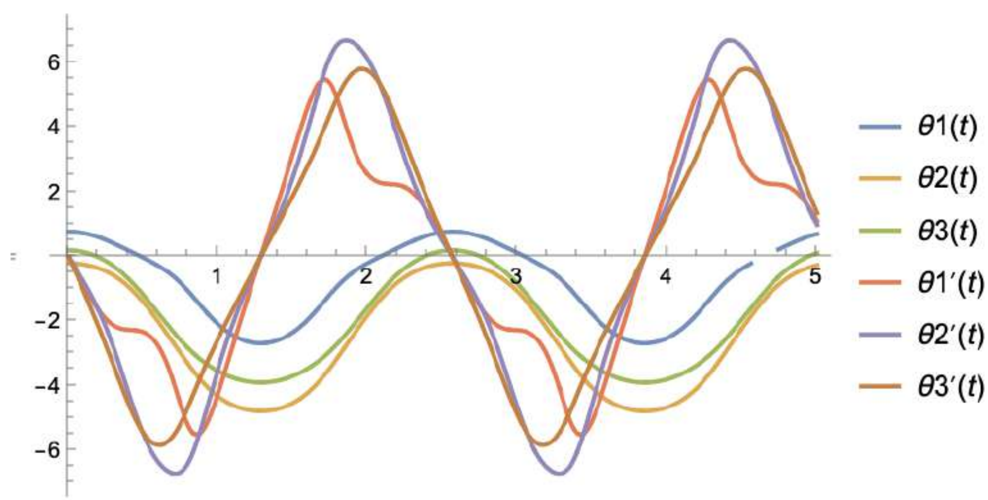


Figure 4.1: Forward dynamics simulation: joint positions and velocities

To validate the physical consistency of the dynamic formulation, the total mechanical energy is computed at every time step as:

$$TE(t) = T(t) + V(t), \quad \Delta E = \frac{TE(t) - TE(0)}{TE(0)} \times 100\% \tag{4.10}$$

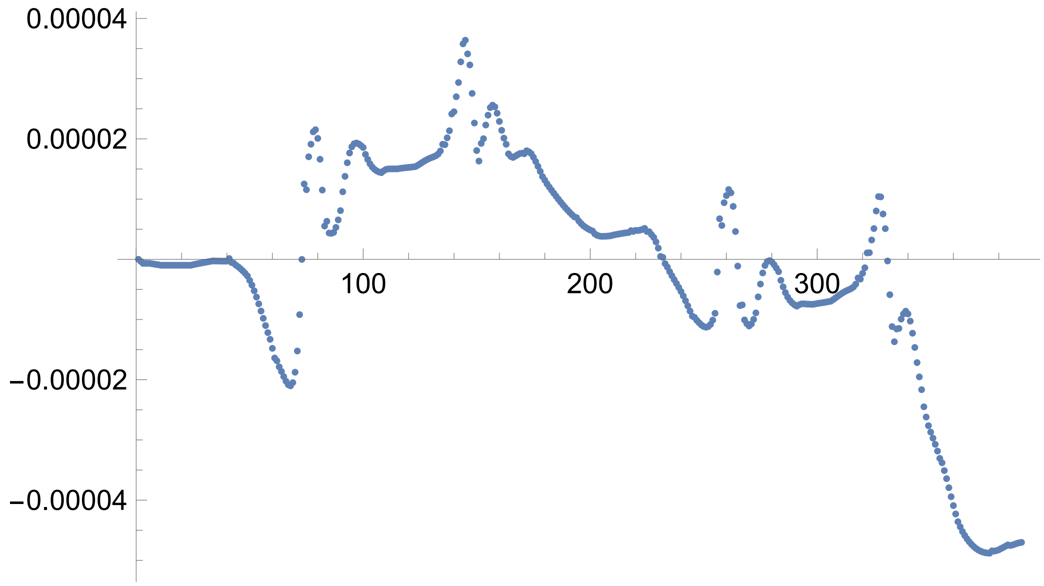


Figure 4.2: Total energy conservation during simulation

4.3 Trajectory Definition and Inverse Kinematics

The trajectory planning begins by specifying a circular path for the coupler point of the four-bar linkage. The path is defined parametrically by a time-dependent angular variable $\phi(t)$:

$$x_d(t) = x_c + R\cos\phi(t), \tag{4.11}$$

$$y_d(t) = y_c + R\sin\phi(t) \tag{4.12}$$

where $(x_c, y_c) = (0.8, 0.6)$ m is the center and $R = 0.26$ m is the radius. To ensure smooth start and stop conditions, $\phi(t)$ is chosen as a cubic polynomial satisfying zero velocity at the endpoints:

$$\phi(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{4.13}$$

with boundary conditions:

$$\phi(0) = 0, \quad \phi(t_f) = 2\pi, \quad \dot{\phi}(0) = 0, \quad \dot{\phi}(t_f) = 0 \tag{4.14}$$

for $t_f = 5$ s.

Solving this system yields:

$$\phi(t) = 0.7540\,t^2 - 0.1005\,t^3 \tag{4.15}$$

The time derivatives are:

$$\dot{\phi}(t) = 1.508\,t - 0.3015\,t^2, \tag{4.16}$$

$$\ddot{\phi}(t) = 1.508 - 0.603\,t \tag{4.17}$$

These are used to derive the linear velocities and accelerations of the end-effector point:

$$\dot{x}_d(t) = -R\dot{\phi}(t)\sin\phi(t), \qquad \dot{y}_d(t) = R\dot{\phi}(t)\cos\phi(t), \tag{4.18}$$

$$\ddot{x}_d(t) = -R[\ddot{\phi}(t)\sin\phi(t) + \dot{\phi}^2(t)\cos\phi(t)], \quad \ddot{y}_d(t) = R[\ddot{\phi}(t)\cos\phi(t) - \dot{\phi}^2(t)\sin\phi(t)] \tag{4.19}$$

To compute the joint angles corresponding to this desired path, inverse kinematics is performed. The position of the coupler point as a function of joint angles is given by:

$$x_d = l_1\cos\theta_1 + l_2\cos\theta_2, \tag{4.20}$$

$$y_d = l_1\sin\theta_1 + l_2\sin\theta_2 \tag{4.21}$$

With link lengths:

$$l_1 = 0.148 \text{ m}, \quad l_2 = 0.254 \text{ m}, \quad l_3 = 0.276 \text{ m}, \quad l_0 = 0.4 \text{ m} \tag{4.22}$$

By applying tangent-half angle substitution and solving the resulting quadratic equation, θ_1 is obtained, followed by computation of θ_2 and θ_3 using loop-closure relationships.

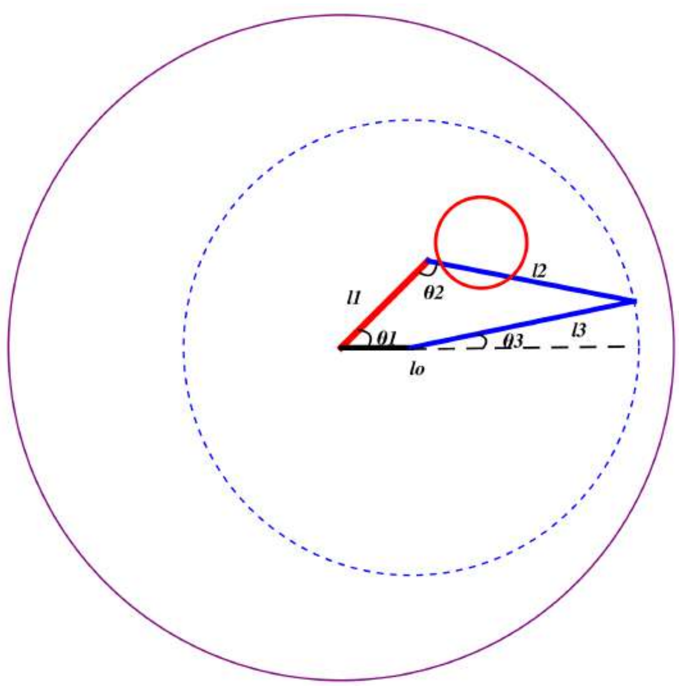


Figure 4.3: Workspace and trajectory planning for a four-bar mechanism

4.4 Inverse Dynamics

Inverse dynamics is used to determine the actuator torques required to achieve the trajectory obtained from simulation. The generalized torque vector $\mathbf{Q}_{\mathrm{nc}}$[?] is computed using:

$$\mathbf{A} = \mathbf{J}_\eta \mathbf{M}^{-1} \mathbf{J}_\eta^T, \tag{4.23}$$

$$\mathbf{B} = \mathbf{I} - \mathbf{J}_\eta^T \mathbf{A}^{-1} \mathbf{J}_\eta \mathbf{M}^{-1}, \tag{4.24}$$

$$\mathbf{Q} = \mathbf{B}^{\#} \left(\mathbf{M}\ddot{q} + \mathbf{J}_\eta^T \mathbf{A}^{-1} \dot{\mathbf{J}}_\eta \dot{q} \right) + \mathbf{G}, \tag{4.25}$$

$$\mathbf{Q}_{\mathrm{nc}} = \mathbf{Q} + \mathbf{E}\mathbf{h} \tag{4.26}$$

Here, $\mathbf{E} = \mathbf{I} - \mathbf{B}^{\#}\mathbf{B}$ and $\mathbf{h}$ is an arbitrary vector. In the current implementation, $\mathbf{h} = 0$, and thus $\mathbf{Q}_{\mathrm{nc}} \neq \mathbf{Q}$ in the general case, but they are equivalent under the chosen simplification.

The Lagrange multipliers $\boldsymbol{\lambda}$, representing the constraint forces, are given by:

$$\boldsymbol{\lambda} = -\mathbf{A}^{-1} \left(\dot{\mathbf{J}}_{\eta q} \dot{\mathbf{q}} + \mathbf{J}_{\eta q} \mathbf{a} \right) \tag{4.27}$$

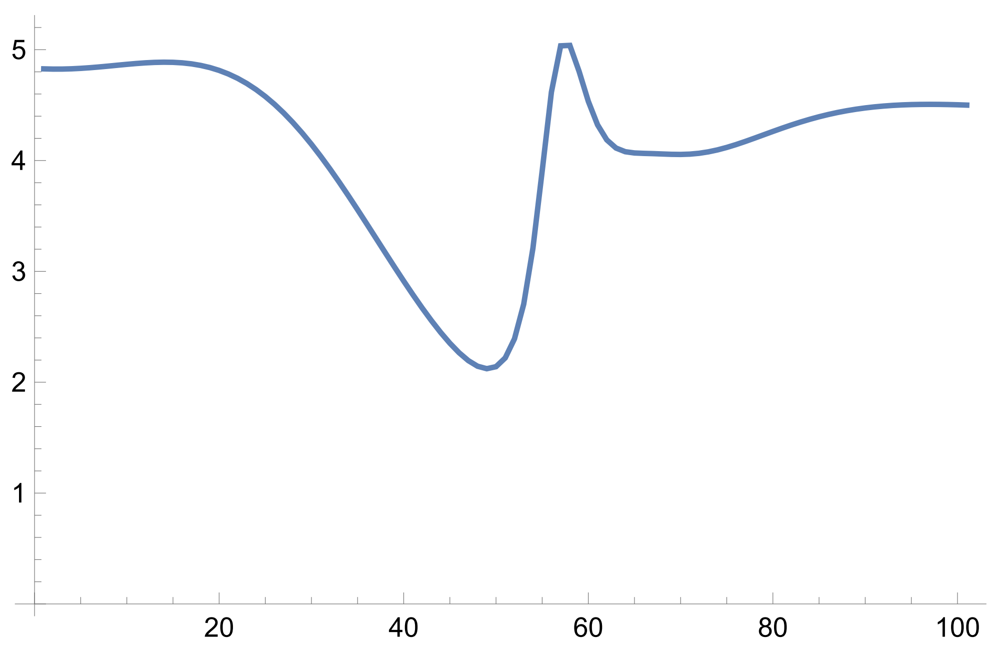


Figure 4.4: Inverse dynamics result: computed actuator torque profile

4.5 Dynamics: Delta Robot

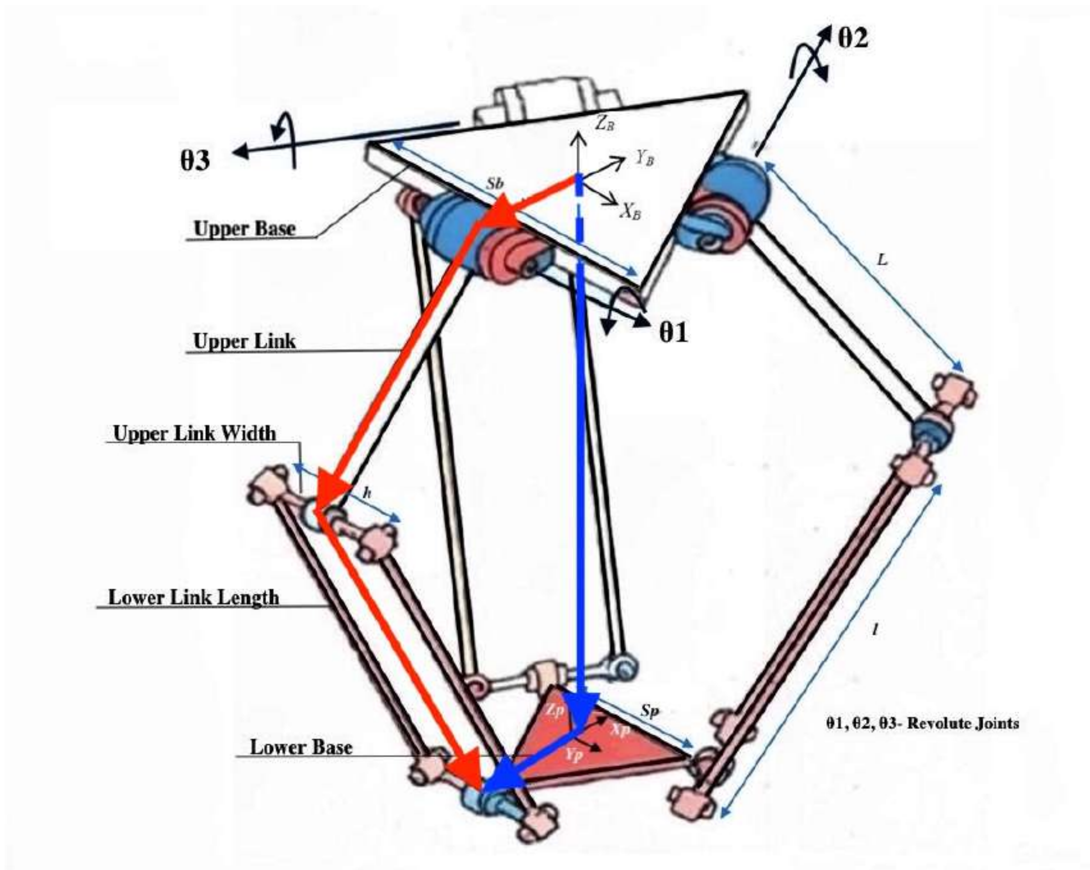


Figure 4.5: Caption

Chapter 5

Fabrication and Assembly

This chapter describes the process of physical realization of a Delta-type parallel robot developed during an internship. The mechanism was fabricated using rapid prototyping techniques, assembled with off-the-shelf components, and iteratively refined over the course of the internship. A combination of CAD modeling, additive manufacturing, and embedded system integration was used to bring the mechanism from concept to functioning prototype.

5.1 Design and Manufacturing Approach

The robot was designed using 3D modeling tools and fabricated using a Raise3D Pro 3D printer with PLA (Polylactic Acid) filament. Additive manufacturing was chosen over conventional subtractive methods due to its ability to rapidly prototype complex geometries with minimal post-processing. Given the limited duration of the internship (approximately one month), this approach allowed for multiple iterations to be tested in a short span of time.

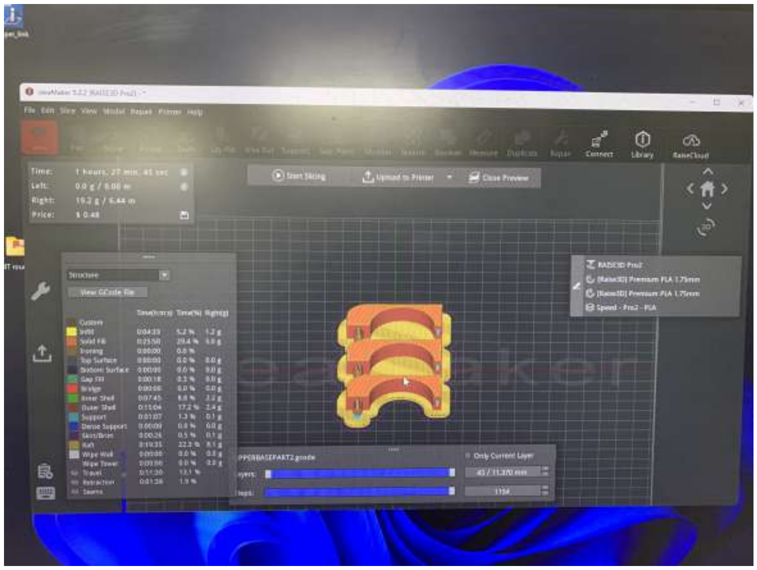


Figure 5.1: Printing process in the Raise3D Pro 3D printer: Slicing a component in the ideaMaker software showing the layered toolpath, estimated print time (1 hour, 27 minutes), and material consumption.

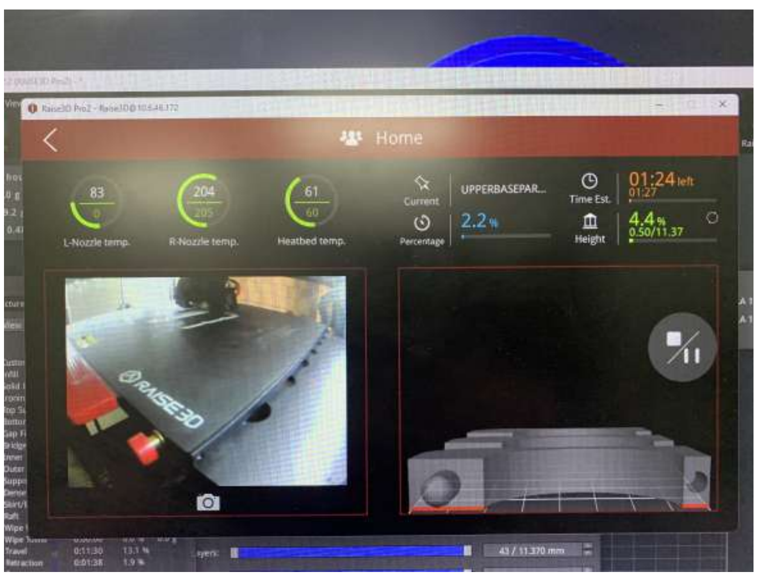


Figure 5.2: Printing process in the Raise3D Pro 3D printer: The printer's control interface at the start of a print job, displaying critical parameters like nozzle and heatbed temperatures alongside a live camera feed of the build plate.

The structure, consisting of the base, arms, and moving platform, was printed as modular components. After each iteration, necessary modifications were made to address issues such as structural stability, joint clearances, and tolerances. This iterative design–print–test loop, illustrated by the printing process shown in Figure 5.3, was essential to arrive at a functional and stable assembly.

Figure 5.3: Printing process in the Raise3D Pro 3D printer: A close-up view of the first layer being deposited on the build plate, marking the beginning of the physical fabrication of a component.

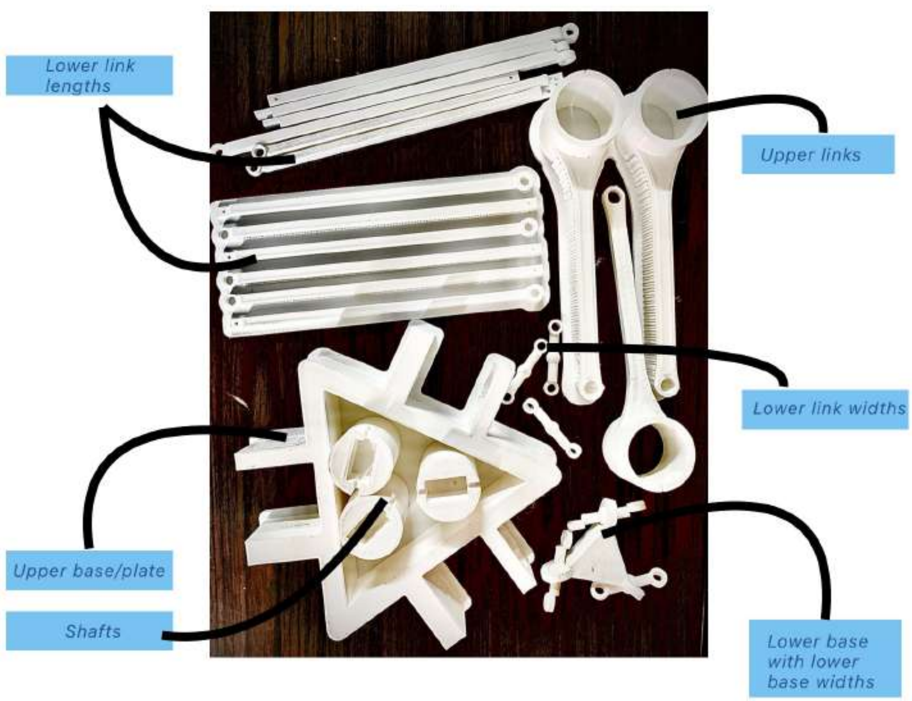


Figure 5.4: Fabricated Delta robot frame using 3D printed PLA components

5.2 Actuation and Control Integration

The final robot was actuated using three Dynamixel AX-12A smart servo motors, each controlled by a dedicated Arduino Uno board. The Arduinos received commands via serial communication and issued position signals to the respective Dynamixels. The motors were mounted symmetrically at the top platform and connected to the arms through printed servo brackets.

- **Motors used**: 3 × Dynamixel AX-12A
- **Microcontrollers**: 3 × Arduino Uno (one per actuator)
- **Control interface**: USB serial with position command protocols

Despite the modular control setup, several challenges were encountered, particularly due to latency and synchronization issues in serial communication. Since the Dynamixels operate in a half-duplex mode and the Arduinos were independently controlling each joint, achieving real-time synchronization required careful calibration and optimization of command timing.

5.3 Challenges and Iterations

The fabrication and assembly process was marked by multiple iterations and design modifications. Some of the major challenges encountered included:

- **Link misalignments and clearances**: Initial designs had tolerances that were either too tight (leading to binding) or too loose (resulting in backlash).
- **Servo bracket fatigue**: Due to repeated disassembly during prototyping, the printed joints required reinforcement in later stages.
- **Communication latency**: Serial delay between the control unit and multiple Arduino boards introduced difficulty in coordinated motion, requiring low-level optimization and simplified testing.
- **Workspace mismatch**: Early prototypes showed a reduced effective workspace compared to simulation due to mechanical constraints and limited range of the AX-12A actuators.

Each issue led to design improvements, such as adding end-stop reinforcements, widening joint tolerances, or adjusting the inverse kinematic mapping to fit real actuator ranges.

Figure 5.5: Assembled robot undergoing pose validation and calibration

5.4 Additive Manufacturing vs. Traditional Techniques

The use of 3D printing for fabrication brought notable advantages in the context of this short-term project:

- **Rapid iteration**: Designs could be updated and reprinted within hours, enabling fast prototyping cycles.
- **Complex geometry fabrication**: Non-planar surfaces and integrated features (e.g., servo slots) were easy to produce without complex fixtures.
- **Material efficiency**: PLA provided a lightweight solution suitable for moderate-load applications.

However, compared to traditional CNC-machining or metal fabrication, some limitations were observed:

- **Lower mechanical strength**: PLA parts showed visible deformation under continuous dynamic loading, especially at arm joints.
- **Surface finish and precision**: Post-processing was sometimes required to ensure proper assembly fits.
- **Thermal sensitivity**: High ambient temperatures could affect the dimensional stability of printed parts.

Despite these trade-offs, the flexibility and speed of additive manufacturing made it the most viable choice for a constrained-duration internship, where the primary goal was to validate functional motion and control, rather than long-term robustness.

Summary

The fabrication and assembly of the Delta-type robot exemplified the benefits and challenges of building a parallel manipulator from scratch within a tight timeline. Through continuous iteration and the use of modern fabrication tools, a working prototype was realized and integrated with control systems. The experience highlighted the need for tight coupling between design, fabrication, and testing processes in hardware-based robotics projects.

Chapter 6

Conclusion

This work brought together multiple facets of robotics — from mathematical modeling to physical prototyping — in an effort to understand and implement two widely studied mechanisms: the four-bar linkage and the Delta parallel robot. What began as symbolic equations and constraint formulations eventually evolved into a functioning, controllable robotic system, showcasing the complete pipeline from theory to practice.

For the four-bar mechanism, we explored how even a relatively simple planar system can exhibit complex behavior when analyzed dynamically. Trajectory planning was implemented using smooth cubic polynomial interpolation, and the motion was verified through both forward and inverse dynamics. The use of constrained Euler-Lagrange methods allowed us to precisely model the system's dynamics and compute torque requirements while ensuring the kinematic constraints were consistently satisfied. These results, supported by numerical simulations, formed a solid foundation for understanding constrained multibody systems.

The Delta robot added another layer of complexity and interest. Its parallel structure and pure translational motion required careful consideration of actuator-space formulations and Jacobian-based constraint modeling. Using symbolic computation, we derived the necessary dynamic equations and simulated torque profiles corresponding to predefined end-effector trajectories. The robot was then brought to life through a CAD design modeled in `Fusion 360`, 3D-printed with PLA material, and actuated using three Dynamixel AX-12A motors, each controlled through Arduino Uno boards. The physical build demanded multiple design iterations and presented challenges ranging from mechanical clearances to synchronization delays — all of which were addressed through experimentation and adaptation.

What truly tied this project together was the ability to connect deep theoretical modeling with hands-on hardware implementation. Simulations helped us understand behaviors before physical testing, and symbolic dynamics gave us insights into energy flow and constraint enforcement. Despite the limited duration of the internship, the end result was a working robotic system backed by solid computation and clean design.

What This Work Achieved:

- Developed a complete dynamics framework for a four-bar mechanism, including kinematics, trajectory planning, and torque estimation.
- Modeled the actuator-space dynamics of a Delta robot using symbolic Lagrangian formulation with constraint Jacobians.
- Designed and fabricated a functioning Delta robot using additive manufacturing techniques.
- Bridged simulation results with physical implementation, validating torque and motion profiles.
- Overcame real-world issues like serial communication latency and mechanical alignment through iteration and testing.

In essence, this project reflects the essence of engineering — using equations and principles to build something real. It reinforces the idea that theoretical modeling is not just an academic exercise, but a powerful tool that, when combined with hardware intuition and iteration, can lead to working, meaningful systems. The skills, tools, and insights gained through this journey form a strong foundation for future work in robotics, whether in advanced control, automation, or machine design.

References

[1] R. L. W. II, "Delta robot kinematics," 2003.

[2] M. Laribi, L. Romdhane, and S. Zeghloul, "Analysis and dimensional synthesis of the delta robot for a prescribed workspace," *Mechanism and Machine Theory*, vol. 42, no. 7, pp. 859–870, 2007.

Appendix: Arduino Control Codes

This appendix includes the motor control Arduino sketches used for testing and controlling the Delta and four-bar mechanisms.

Motor Control Code for 3 Motors

Listing 1: motor3control.ino

```
#include "Arduino.h"
#include "AX12A.h"

#define DirectionPin  (10u)
#define BaudRate      (1000000ul)
#define ID            (3u)

int target_pos = 512; // Initial position
const int MIN_POS = 0;   // Minimum position for AX-12A
const int MAX_POS = 1023; // Maximum position for AX-12A

String inputString = ""; // String to hold incoming serial data
bool stringComplete = false; // Flag for completed input

void setup()
{
  Serial.begin(BaudRate); // Start Serial at AX-12A baud rate
  ax12a.begin(BaudRate, DirectionPin, &Serial); // Initialize AX-12A on same Serial
  inputString.reserve(10); // Reserve space for input string
  ax12a.move(ID, target_pos); // Move to initial position
  delay(100); // Wait for motor to stabilize
  Serial.begin(9600); // Switch to Serial Monitor baud rate
  Serial.println("Enter a position (0 to 1023 or negative for reverse):");
}

void loop()
{
  // Read input from Serial Monitor
  while (Serial.available()) {
    char inChar = (char)Serial.read();
    if (inChar == '\n') { // Check for newline to complete the input
      stringComplete = true;
```

```
      break;
    } else {
      inputString += inChar; // Add character to input string
    }
  }

  // Process input if complete
  if (stringComplete) {
    // Switch Serial to AX-12A baud rate
    Serial.end();
    Serial.begin(BaudRate);

    // Convert string to integer
    int new_pos = inputString.toInt();

    // Validate the position
    if (new_pos >= MIN_POS && new_pos <= MAX_POS) {
      target_pos = new_pos;
      ax12a.move(ID, target_pos); // Move motor to the target position
      delay(50); // Allow motor to move
    }

    // Switch back to Serial Monitor baud rate
    Serial.end();
    Serial.begin(9600);

    // Provide feedback
    if (new_pos >= MIN_POS && new_pos <= MAX_POS) {
      Serial.print("Moving to position: ");
      Serial.println(target_pos);
    } else {
      Serial.println("Invalid position! Enter a value between 0 and 1023.");
    }

    // Clear the string and flag
    inputString = "";
    stringComplete = false;

    Serial.println("Enter a new position:");
  }
}
```

Motor Control Code for 4 Motors

Listing 2: motor4control.ino

```
#include "Arduino.h"
#include "AX12A.h"

#define DirectionPin  (10u)
#define BaudRate      (1000000ul)
#define ID            (4u)

int target_pos = 512; // Initial position
const int MIN_POS = 0;   // Minimum position for AX-12A
const int MAX_POS = 1023; // Maximum position for AX-12A


String inputString = ""; // String to hold incoming serial data
bool stringComplete = false; // Flag for completed input

void setup()
{
  Serial.begin(BaudRate); // Start Serial at AX-12A baud rate
  ax12a.begin(BaudRate, DirectionPin, &Serial); // Initialize AX-12A on same Serial
  inputString.reserve(10); // Reserve space for input string
  ax12a.move(ID, target_pos); // Move to initial position
  delay(100); // Wait for motor to stabilize
  Serial.begin(9600); // Switch to Serial Monitor baud rate
  Serial.println("Enter a position (0 to 1023 or negative for reverse):");
}

void loop()
{
  // Read input from Serial Monitor
  while (Serial.available()) {
    char inChar = (char)Serial.read();
    if (inChar == '\n') { // Check for newline to complete the input
      stringComplete = true;
      break;
    } else {
      inputString += inChar; // Add character to input string
    }
```

```
  }

  // Process input if complete
  if (stringComplete) {
    // Switch Serial to AX-12A baud rate
    Serial.end();
    Serial.begin(BaudRate);

    // Convert string to integer
    int new_pos = inputString.toInt();

    // Validate the position
    if (new_pos >= MIN_POS && new_pos <= MAX_POS) {
      target_pos = new_pos;
      ax12a.move(ID, target_pos); // Move motor to the target position
      delay(50); // Allow motor to move
    }

    // Switch back to Serial Monitor baud rate
    Serial.end();
    Serial.begin(9600);

    // Provide feedback
    if (new_pos >= MIN_POS && new_pos <= MAX_POS) {
      Serial.print("Moving to position: ");
      Serial.println(target_pos);
    } else {
      Serial.println("Invalid position! Enter a value between 0 and 1023.");
    }

    // Clear the string and flag
    inputString = "";
    stringComplete = false;

    Serial.println("Enter a new position:");
  }
}
```

Motor Control Code for 6 Motors

Listing 3: motor6control.ino

```cpp
#include "Arduino.h"
#include "AX12A.h"

#define DirectionPin  (10u)
#define BaudRate      (1000000ul)
#define ID            (6u)

int target_pos = 512; // Initial position
const int MIN_POS = 0;   // Minimum position for AX-12A
const int MAX_POS = 1023; // Maximum position for AX-12A

String inputString = ""; // String to hold incoming serial data
bool stringComplete = false; // Flag for completed input

void setup()
{
  Serial.begin(BaudRate); // Start Serial at AX-12A baud rate
  ax12a.begin(BaudRate, DirectionPin, &Serial); // Initialize AX-12A on same Serial
  inputString.reserve(10); // Reserve space for input string
  ax12a.move(ID, target_pos); // Move to initial position
  delay(100); // Wait for motor to stabilize
  Serial.begin(9600); // Switch to Serial Monitor baud rate
  Serial.println("Enter a position (0 to 1023 or negative for reverse):");
}

void loop()
{
  // Read input from Serial Monitor
  while (Serial.available()) {
    char inChar = (char)Serial.read();
    if (inChar == '\n') { // Check for newline to complete the input
      stringComplete = true;
      break;
    } else {
      inputString += inChar; // Add character to input string
    }
  }

  // Process input if complete
  if (stringComplete) {
    // Switch Serial to AX-12A baud rate
```

```
    Serial.end();
    Serial.begin(BaudRate);

    // Convert string to integer
    int new_pos = inputString.toInt();

    // Validate the position
    if (new_pos >= MIN_POS && new_pos <= MAX_POS) {
      target_pos = new_pos;
      ax12a.move(ID, target_pos); // Move motor to the target position
      delay(50); // Allow motor to move
    }

    // Switch back to Serial Monitor baud rate
    Serial.end();
    Serial.begin(9600);

    // Provide feedback
    if (new_pos >= MIN_POS && new_pos <= MAX_POS) {
      Serial.print("Moving to position: ");
      Serial.println(target_pos);
    } else {
      Serial.println("Invalid position! Enter a value between 0 and 1023.");
    }

    // Clear the string and flag
    inputString = "";
    stringComplete = false;

    Serial.println("Enter a new position:");
  }
}
```

Working Version 1

Listing 4: working1.ino

```
/*
 * Example showing how to send position commands to AX-12A
 */

#include "Arduino.h"
```

```
#include "AX12A.h"

#define DirectionPin    (10u)
#define BaudRate                      (1000000ul)
#define ID                                    (3u)

int initial_pos = 512;
int max = initial_pos ;
int min = initial_pos - 200;

int pos = initial_pos;
int delta = 5;

void setup()
{
        ax12a.begin(BaudRate, DirectionPin, &Serial);
}

void loop()
{
        pos = pos + delta;

        if (pos > max)
        {
                pos = max;
                delta *= -1;
        }

        if (pos < min)
        {
                pos = min;
                delta *= -1;
        }

        ax12a.move(ID, pos);
        delay(20);
}
```

Working Version 2

Listing 5: working2.ino

```
/*
 * Example showing how to send position commands to AX-12A
 */

#include "Arduino.h"
#include "AX12A.h"

#define DirectionPin    (10u)
#define BaudRate                    (1000000ul)
#define ID                                  (4u)

int initial_pos = 512;
int max = initial_pos + 200 ;
int min = initial_pos  ;

int pos = initial_pos;
int delta = 5;

void setup()
{
        ax12a.begin(BaudRate, DirectionPin, &Serial);
}

void loop()
{
        pos = pos + delta;

        if (pos > max)
        {
                pos = max;
                delta *= -1;
        }

        if (pos < min)
        {
                pos = min;
                delta *= -1;
        }

        ax12a.move(ID, pos);
        delay(20);
```

```
}
```

Working Version 3

Listing 6: working3.ino

```
/*
 * Example showing how to send position commands to AX-12A
 */

#include "Arduino.h"
#include "AX12A.h"

#define DirectionPin    (10u)
#define BaudRate                    (1000000ul)
#define ID                                  (6u)

int initial_pos = 512;
int max = initial_pos + 150;
int min = initial_pos ;

int pos = initial_pos;

int delta = 5;

void setup()
{
        ax12a.begin(BaudRate, DirectionPin, &Serial);
}

void loop()
{
        pos = pos + delta;

        if (pos > max)
        {
                pos = max;
                delta *= -1;
        }

        if (pos < min)
        {
                pos = min;
                delta *= -1;
```

```
        }

        ax12a.move(ID, pos);
        delay(20);
}
```