DE-41 (DME)

NUST COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

PATH PLANNING OF ROBOTIC MANIPULATOR FOR APPLICATION OF AUTOMATED LAYUP IN COMPOSITE MANUFACTURING

A PROJECT REPORT

DE-41 (DME)

Submitted by

MUHAMMAD TARIQ SHAH

NUFAIL AHMED NADEEM

FARHAN ALI

BACHELORS IN MECHANICAL ENGINEERING

YEAR

2023

PROJECT SUPERVISOR

DR. RAJA AMER AZIM

PROJECT CO-SUPERVISOR

DR. AMIR HAMZA

NUST COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING
PESHAWAR ROAD, RAWALPINDI

YEAR 2023

DECLARATION

We hereby declare that no portion of the work referred to in this Project Thesis has been submitted in support of an application for another degree or qualification of this of any other university or other institute of learning. If any act of plagiarism is found, we are fully responsible for every disciplinary action taken against us depending upon the seriousness of the proven offence, even the cancellation of our degree.

COPYRIGHT STATEMENT

ACKNOWLEDGEMENTS

First, we are very thankful to Almighty Allah who made our journey from none without any source to higher education from a prestigious university. We really want to show our huge gratitude to Pakistan, its wonderful people, and the National University of Science and Technology, whose financial support made our dream of higher education, true because we did not have the good fortune at all to afford higher education. We are grateful to the Almighty Allah, the most merciful who bestowed us with such a wonderful project, full of learning and an opportunity to automate our manufacturing industry. Words are not enough to thank our supervisor Dr. Raja Amir and our Co-Supervisor for their consistent guidance and mentorship, due to which we were able to achieve this huge milestone. We are also very humbled to Research Assistants Umer Anjum, Wajih Khan and Team Lead RML Hamza Ahmed.

ABSTRACT

The project achieved the automation of the rolling process for the application of automated layup, specifically for the rolling of the pre-preg layer. However, it is important to note that the actual laying of the layers and stacking was still done manually (although the ultimate objective of the project is to fully automate the layup process with three collaborative robots). The project includes the design and construction of a rolling-end effector, which was not the primary objective of the project.

To analyze the feasibility and efficiency of the automated layup application, the project extensively conducted software simulations using ROS, MoveIt, and RoboDK. Various path planning algorithms, including A*, RRT, and Dijkstra, were tested, and valuable insights were gained from the simulation results. These findings enhanced the project's understanding of the capabilities of the algorithms and contributed to improved robotic control throughout the design process. The project compared the performance of the algorithms based on planning time, collision avoidance, and overall trajectory optimization.

Furthermore, the project emphasized the integration of path planning algorithms into real-world hardware, specifically the UR5 manipulators. This integration process involved evaluating the robots, optimizing the algorithms, and ensuring smooth communication between the robot controller and the system software. Rigorous testing and iterations were conducted during this phase to address any discrepancies between the simulations and the execution in the real-world environment.

TABLE OF CONTENTS

DECLARATION **2**
COPYRIGHT STATEMENT **2**
ACKNOWLEDGEMENTS **3**
ABSTRACT **4**
LIST OF FIGURES **6**
LIST OF TABLES **6**
Chapter 01: Introduction **7**
1.1 Composite Materials 7
1.1.1 Pre-Preg 7
1.2 Composite Manufacturing 7
1.3 Manual Layup 8
1.4 Background on Robotic Manipulators: 9
1.5 Application of Robotic Manipulators in Composite Manufacturing: 9
1.6 Literature Review 10
1.6.1 Path Planning for Robotic Manipulators 10
1.6.2 Path Planning in Automated Layup for Composite Manufacturing 11
1.6.3 Limitations and Gaps in Existing Research 11
1.6.4 Addressing Limitations in Our Project 11
1.7 Problem Statement 11
1.8 Deliverables 12
1.9 Sustainable Development Goals 12
Chapter 02: Mathematical Modeling **14**
2.1 Forward Kinematics: 14
2.2 Inverse Kinematics: 14
2.3 Denavit–Hartenberg (DH) parameters of UR5 Robot: 15
2.4 Transformation Matrix: 15
2.5 Universal Robot 5 (UR 5) 16
2.6 Skateboard Manufacturing 17
2.7 Skateboard Design 18
Chapter 03: Implementation of Algorithm **20**
3.1 Factors affecting the selection of Path Planning 20
3.2 Types of Algorithms: 21
3.3 Methodology: 22
3.4 Rapidly exploring Random Trees (RRT): 23
3.5 Types of RRT 25
Chapter 04: Software Simulations and Results **27**
4.1 Fibersim and Optimum Layup Strategy: 27
4.2 Implementation of ROS, MoveIt, and RQT for Path Planning Evaluation: 28
4.3 Integration of Path Planning Algorithms into UR5 Manipulators for Real-World Hardware: 30
4.4 Inverse Kinematics Solutions for 6 DOF (RoboAnalyzer): 31
4.5 UR5 Inverse Kinematics Results: 32
Chapter 05: Fabrication **38**
5.1 Materials Selection 38

5.2 Preparation of Prepreg 42
5.3 Automated rolling in Layup of Prepreg: 43
5.4 Curing Process: 44
REFERENCES 45
APPENDIX I 46

LIST OF FIGURES

Figure 1: A 3-D layout of a composite material 7
Figure 2: Hand Layup Process for Prepreg Man 9
Figure 3: Sustainable Development Goals 13
Figure 4: DH parameters of UR5 Robot 15
Figure 5: A UR-5 Robot 16
Figure 6: Skateboard 3D View 18
Figure 7: Skateboard Front View 19
Figure 8: Skateboard Top View 19
Figure 9: Key points about the significance of RRT 24
Figure 10: RRT Algorithm 25
Figure 11: RRT-Connect Algorithm Description 26
Figure 12: Bi-Directional RRT Algorithm 26
Figure 13 Fibersim path feasibility 28
Figure 14 Fibersim Path feasibility 28
Figure 15 Integrating UR5 with MoveIt 29
Figure 16 Real time MoveIt simulation 31
Figure 17: RoboAnalyzer End Effector Position 31
Figure 18 RoboAnalyzer Twist Angles 31
Figure 19: Inverse Kinematics Results for Point 1 32
Figure 20: Inverse Kinematics Results for Point 2 32
Figure 21: Inverse Kinematics Results for Point 3 33
Figure 22: Inverse Kinematics Results for Point 4 33
Figure 23: Inverse Kinematics Results for Point 5 34
Figure 24: Inverse Kinematics Results for Point 34
Figure 25: Inverse Kinematics Results for Point 7 35
Figure 26: Inverse Kinematics Results for Point 8 35
Figure 27: Inverse Kinematics Results for Point 9 36
Figure 28: Inverse Kinematics Results for Point 10 36
Figure 29: Inverse Kinematics Results for Point 11 37
Figure 30: RoboDK Simulation of UR 37
Figure 31: Mixing of Epoxy Resin 42
Figure 33: Prepreg sheets ready for layup 42
Figure 34: Automated Layup using UR5. 43
Figure 35: Oven arrangements for curing of prepreg 44

LIST OF TABLES

Table 1: Technical specification of UR5 robot arm 17

Chapter 01: Introduction

1.1 Composite Materials

A composite material (also called a composition material or shortened to composite, which is the common name) is a material which is produced from two or more constituent materials.[1] These constituent materials have notably dissimilar chemical or physical properties and are merged to create a material with properties unlike the individual elements. Within the finished structure, the individual elements remain separate and distinct, distinguishing composites from mixtures and solid solutions.

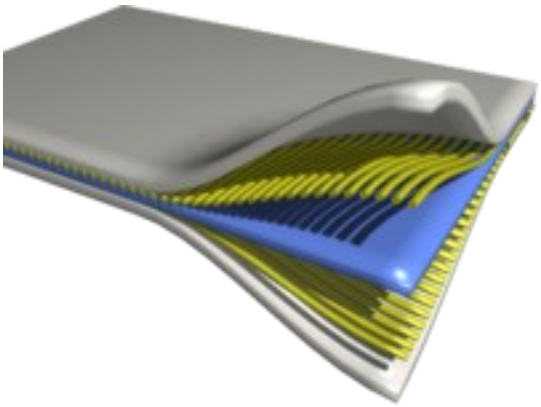


Figure 1: A 3-D layout of a composite material

1.1.1 Pre-Preg

Pre-preg is a composite material made from "pre-impregnated" fibers and a partially cured polymer matrix, such as epoxy or phenolic resin, or even thermoplastic mixed with liquid rubbers or resins. The fibers often take the form of a weave, and the matrix is used to bond them together and to other components during manufacture.

1.2 Composite Manufacturing

Composite materials are widely used in various industries due to their exceptional strength-to-weight ratio, corrosion resistance, and design flexibility. In the context of this project on path planning for robotic manipulators in automated layup, it is essential to understand the different types of composite manufacturing processes. Some common types include:

Hand Layup: Hand layup is a manual process where composite materials, such as fibers and resin, are carefully placed and consolidated by hand onto a mold or tool surface. Although labor-intensive,

hand layup allows for flexibility in material placement and is suitable for small-scale production or complex-shaped components.

Automated Tape Laying (ATL): ATL involves the use of a machine to precisely position and consolidate composite tapes onto a mold or tool surface. This process offers increased accuracy and repeatability, making it suitable for medium to high-volume production.

Automated Fiber Placement (AFP): AFP is a similar process to ATL but involves the precise placement of continuous fibers rather than tapes. AFP machines can lay down fibers in various orientations and create complex composite structures with optimized strength and stiffness properties.

Resin Transfer Molding (RTM): RTM is a closed-mold process where dry fiber preforms are placed in a mold, and resin is injected under pressure. The resin fills the mold and impregnates the fibers, resulting in a fully consolidated composite part. RTM offers good repeatability and is commonly used for medium to large-sized components.

1.3 Manual Layup

Hand layup is a commonly used process for making composite structures from several plies of carbon-fiber prepreg. The process involves multiple human operators manipulating and conforming layers of prepreg to a mold. The manual layup process is ergonomically challenging, tedious, and limits throughput. Moreover, different operators may perform the process differently, and hence introduce inconsistency.

The manual lay-up of prepreg on a mold was performed in a workshop conducted in the Department of Mechatronics Engineering.

We are working to automate the process of manual layup with the help of collaborative robot.

Figure 2: Hand Layup Process for Prepreg Man

1.4 Background on Robotic Manipulators:

Robotic manipulators have emerged as crucial tools in various industrial applications, including composite manufacturing. Composite materials, such as carbon fiber-reinforced polymers (CFRP), offer exceptional strength-to-weight ratios, making them ideal for aerospace, automotive, and other high-performance industries. However, the manual layup process for composite manufacturing is labor-intensive, time-consuming, and prone to errors. To address these challenges, automated layup using robotic manipulators has gained significant attention.

Robotic manipulators, commonly referred to as robot arms, are mechanical systems composed of multiple rigid links interconnected by joints. These systems possess the ability to perform precise and repetitive tasks with high accuracy and speed. In the context of composite manufacturing, robotic manipulators offer several advantages over manual methods. They enable the automation of complex tasks such as laying up composite materials on molds, performing precise cuts, and applying consistent pressure during curing processes.

1.5 Application of Robotic Manipulators in Composite Manufacturing:

The application of robotic manipulators in composite manufacturing has led to increased productivity, improved quality control, and reduced labor costs. By automating the layup process, manufacturers can achieve higher repeatability and accuracy, resulting in consistently high-quality composite parts. Furthermore, robotic manipulators can handle complex shapes and intricate patterns, allowing to produce customized composite components.

One of the critical aspects in utilizing robotic manipulators for automated layup in composite manufacturing is path planning. Path planning involves determining the optimal trajectory for the robot arm to follow while avoiding obstacles and ensuring collision-free movements. Efficient path planning algorithms are essential to minimize the time required for the layup process and to optimize material usage.

By implementing path planning algorithms specifically tailored for robotic manipulators, manufacturers can achieve efficient and reliable automated layup. These algorithms consider factors such as the shape and geometry of the composite parts, tooling constraints, and collision avoidance, allowing the robot arm to navigate complex paths accurately and swiftly.[2]

In conclusion, the application of robotic manipulators in composite manufacturing has revolutionized the industry by automating labor-intensive processes and improving overall efficiency. The integration of path planning algorithms further enhances the capabilities of robotic manipulators by ensuring optimized trajectories and collision-free movements. As a result, manufacturers can achieve higher productivity, increased precision, and consistent quality in the production of composite components.

1.6 Literature Review

In this section, we review the existing literature and research related to path planning for robotic manipulators and automated layup in composite manufacturing. The literature search aimed to identify various algorithms and techniques used for path planning in similar applications. Additionally, we aimed to highlight the limitations or gaps in the existing research and explain how our project aims to address them.

1.6.1 Path Planning for Robotic Manipulators

Robotic manipulators play a vital role in numerous industrial applications, and path planning is a critical aspect of their successful operation. Several path planning algorithms have been proposed in the literature to address the challenges of planning collision-free trajectories for robotic manipulators. Notable algorithms include the Rapidly exploring Random Tree (RRT) algorithm proposed by LaValle in 1998 [3] and its variants such as RRT*, RRT-Connect, and Bidirectional RRT. These algorithms have demonstrated effectiveness in navigating complex environments while considering constraints such as joint limits and collision avoidance.

1.6.2 Path Planning in Automated Layup for Composite Manufacturing

In the context of composite manufacturing, automated layup using robotic manipulators requires careful path planning to ensure accurate and efficient placement of composite materials. Several studies have focused on path planning for automated layup processes. For example, Liu et al. proposed a path planning algorithm based on potential fields and B-spline curves for automated layup of composite panels [4]. The algorithm considered the shape and geometry of the composite parts, as well as the tooling constraints, to generate collision-free trajectories.

1.6.3 Limitations and Gaps in Existing Research

While significant progress has been made in path planning for robotic manipulators and automated layup in composite manufacturing, several limitations and gaps persist. One common challenge is the computational complexity of path planning algorithms, especially for large-scale and complex layup tasks. Additionally, the existing algorithms may not adequately consider the specific requirements of composite manufacturing processes, such as the need for smooth and continuous tool paths to avoid defects in the composite structure. Another limitation is the lack of consideration for dynamic obstacles or uncertainties in the environment.

1.6.4 Addressing Limitations in Our Project

Our project aims to address the limitations and gaps in the existing research by proposing a novel path planning algorithm specifically designed for the automated layup of composite parts. Our algorithm combines the advantages of RRT-based algorithms, such as efficient exploration of the configuration space, with customized optimization techniques to generate smooth and continuous tool paths. Furthermore, we incorporate collision detection and avoidance strategies to handle dynamic obstacles in real-time scenarios. Through these advancements, we aim to improve the efficiency, accuracy, and adaptability of the path planning process in composite manufacturing.

1.7 Problem Statement

In this section, we define the problem statement and identify the research gap in the field of path planning for robotic manipulators in the application of automated layup in composite manufacturing. Through an extensive literature review, we have identified the following problem statement and research gap:

The automation of layup processes in composite manufacturing using robotic manipulators presents several challenges, particularly in the domain of path planning. While there have been significant

advancements in path planning algorithms for robotic manipulators, there is a need for more specialized approaches that address the unique requirements and constraints of automated layup in composite manufacturing.

Despite the existing research on path planning for robotic manipulators, there is a research gap in developing algorithms specifically tailored to the application of automated layup in composite manufacturing. Existing algorithms may not consider the intricacies of composite materials, such as the need for precise material placement, avoidance of wrinkling or fiber misalignment, and the requirement for smooth and continuous tool paths. Furthermore, the current algorithms may not adequately account for constraints imposed by the layup process, such as tool orientation, gripper limitations, and collision avoidance in complex environments.

Addressing this research gap is crucial for achieving optimal path planning in automated layup processes, leading to improved accuracy, efficiency, and productivity in composite manufacturing. Developing specialized path planning algorithms that account for the specific requirements and constraints of composite layup can significantly enhance the quality and reliability of composite components while reducing production time and costs.

1.8 Deliverables

We have defined following deliverables for our project:

- Defining optimal path for the application of composite manufacturing
- Defining optimal layup strategy for manufacturing of skateboard.
- Manufacturing of skateboard using automated layup.

1.9 Sustainable Development Goals

In recent years, the concept of sustainable development has gained significant attention worldwide. The United Nations' Sustainable Development Goals (SDGs) provide a comprehensive framework to address global challenges and promote sustainable practices across various sectors. In the context of the project on path planning of a robotic manipulator for automated layup in composite manufacturing, several SDGs are particularly relevant:

Goal 9: Industry, Innovation, and Infrastructure SDG 9 emphasizes the promotion of inclusive and sustainable industrialization, fostering innovation, and building resilient infrastructure. By automating the layup process in composite manufacturing using robotic manipulators, this project aligns with SDG 9 by improving manufacturing efficiency, reducing waste, and enhancing infrastructure for sustainable industrial development.

Goal 12: Responsible Consumption and Production SDG 12 aims to ensure sustainable consumption and production patterns. The project contributes to this goal by automating the layup process, which reduces material waste, minimizes errors, and enables more precise material placement. This, in turn, promotes responsible consumption and production practices in the composite manufacturing industry.

Goal 13: Climate Action SDG 13 calls for urgent action to combat climate change and its impacts. By optimizing the layup process through automated path planning, the project reduces energy consumption and minimizes the carbon footprint associated with manual labour. The use of robotic manipulators in composite manufacturing aligns with SDG 13 by fostering climate-resilient practices and reducing greenhouse gas emissions.

Goal 17: Partnerships for the Goals SDG 17 emphasizes the importance of partnerships and collaboration to achieve sustainable development. The project on path planning for automated layup in composite manufacturing contributes to SDG 17 by fostering collaboration between academia, industry, and other stakeholders. By working together, sharing knowledge, and leveraging expertise, sustainable practices in composite manufacturing can be effectively implemented.

Figure 3: Sustainable Development Goals

Chapter 02: Mathematical Modeling

2.1 Forward Kinematics:

Forward kinematics refers to the process of obtaining position and velocity of end effector, given the known joint angles and angular velocities.

A commonly used convention for selecting frames of reference in robotics applications is the Denavit and Hartenberg (D–H) convention which was introduced by Jacques Denavit and Richard S. Hartenberg. In this convention, coordinate frames are attached to the joints between two links such that one transformation is associated with the joint, [Z], and the second is associated with the link [X]. The coordinate transformations along a serial robot consisting of n links form the kinematics equations of the robot,

where [T] is the transformation locating the end-link.

$T = [Z_1]\ [X_1][Z_2\ ][X_2\ ]\ldots\ [X_{n-1}][Z_n\ X_n]$

where [T] is the transformation locating the end-link.

2.2 Inverse Kinematics:

In computer animation and robotics, inverse kinematics is the mathematical process of calculating the variable joint parameters needed to place the end of a kinematic chain, such as a robot manipulator or animation character's skeleton, in a given position and orientation relative to the start of the chain. Given joint parameters, the position and orientation of the chain's end, e.g., the hand of the character or robot, can typically be calculated directly using multiple applications of trigonometric formulas, a process known as forward kinematics. However, the reverse operation is, in general, much more challenging.

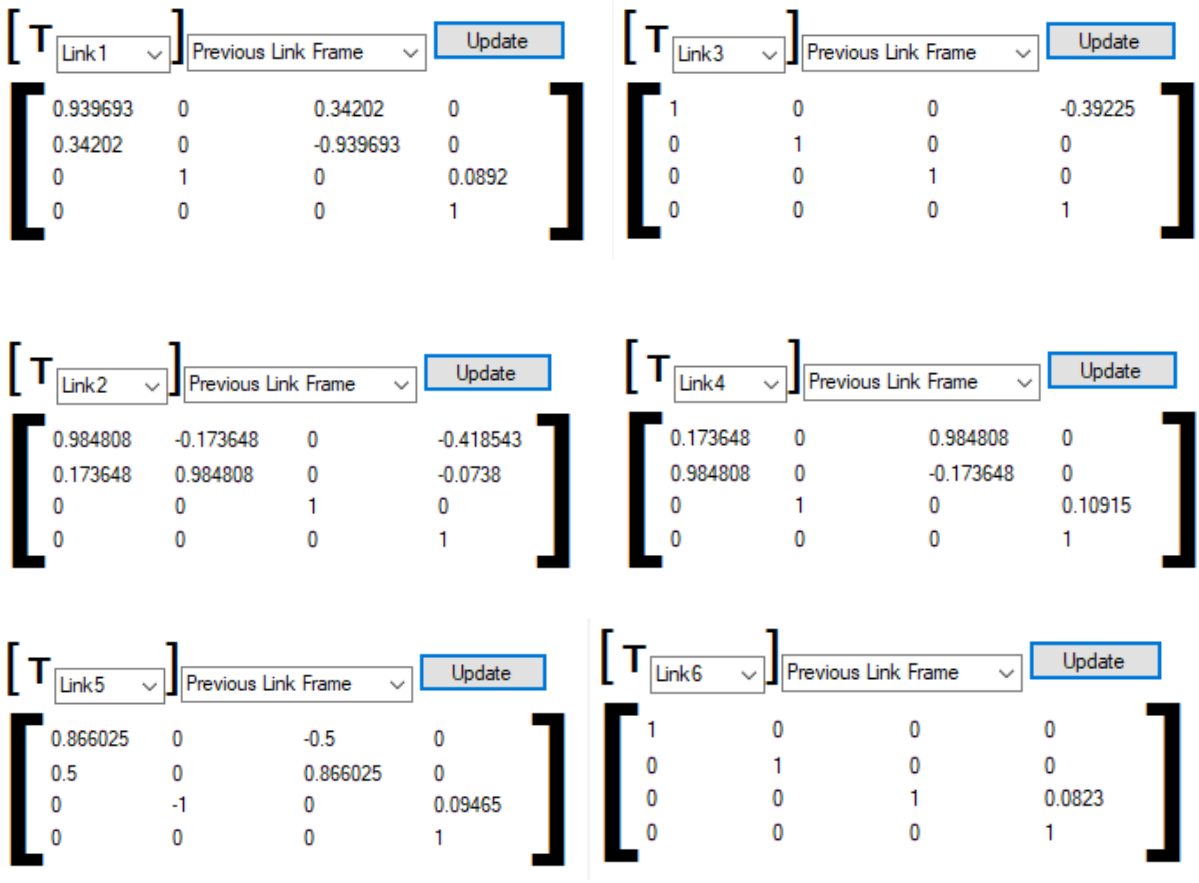
Inverse kinematics is also used to recover the movements of an object in the world from some other data, such as a film of those movements, or a film of the world as seen by a camera which is itself making those movements. This occurs, for example, where a human actor's filmed movements are to be duplicated by an animated character.

2.3 Denavit–Hartenberg (DH) parameters of UR5 Robot:

Joint	q_i [°]	d_i [m]	a_i [m]	α_i [°]	$offset_i$ [°]
Base	q1	0.089159	0	90	0
Shoulder	q2	0	-0.425	0	0
Elbow	q3	0	-0.39225	0	0
Wrist 1	q4	0.10915	0	90	0
Wrist 2	q5	0.09465	0	-90	0
Wrist 3	q6	0.0823	0	0	0

Figure 4: DH parameters of UR5 Robot

2.4 Transformation Matrix:



2.5 Universal Robot 5 (UR 5)

The UR5 is a versatile industrial robot that is light in weight and can handle moderate-duty tasks with great flexibility. It can seamlessly integrate into a wide range of applications. The UR5 robot is well-suited for low-weight collaborative processes such as testing, placing, and picking. With a working radius of up to 33.5 inches or 850mm, the UR5 robot can reach everything with ease, freeing up your employees' time to focus on other important tasks. The Universal Robots UR5 is comprised of three components: the Control Unit, which is the robot's operational center, the Teach Pendant, which is like a tablet with Linux as its operating system, and the Robot Arm.

The robotic arm consists of six revolute joints. In this report these joints will be referred to as Base, Shoulder, Elbow, Wrist1, Wrist2 and Wrist3. The Shoulder and Elbow joint are rotating perpendicular to the Base joint. These three joints relate to long links. The wrist joints control the Tool Center Point (TCP) in the right orientation.[5]

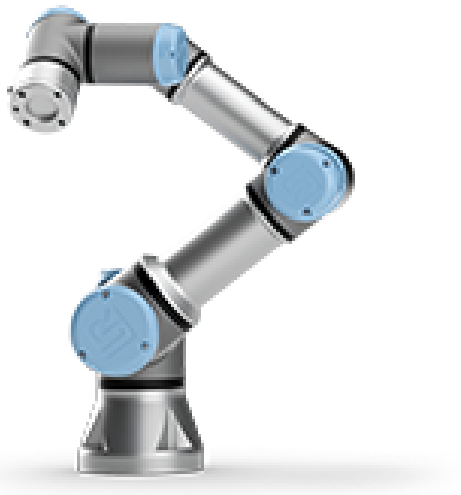


Figure 5: A UR-5 Robot

Weight	18.4 kg
Payload	5 kg
Reach	850 mm
Joint ranges	± 360°
Joint max speed	180° /s
TCP max speed	1 m/s
Degree of freedom	6 rotating joints
Repeatability	± 0.1 mm
I/O Power supply	12 V/24 V 600 mA
Communication	TCP/IP, Ethernet socket & Modbus TCP
Programming	Polyscope graphical user interface
Power consumption	150 W
Power supply	10-240 VAC, 50-50 Hz
Materials	Aluminum, ABS plastic
Temperature	Working range of 0-50°C
Operating life	35000 hours

Table 5.1: Technical specification of UR5 robot arm

2.6 Skateboard Manufacturing

The main aim of our project is to automate the manual layup process with the help of collaborative robot UR5. For this purpose, we have selected a skateboard to implement our path planning algorithm.

Skateboards are generally manufactured from plywood and Al-Wood hybrid. But in case of bending the wood, there are increased chances of defects and cracks.[6]
So, prepreg has been used for the manufacturing of Skateboard using Automated Layup.

2.7 Skateboard Design

Skateboard deck sizes can vary depending on the manufacturer and the specific model. However, there are some common deck sizes that are widely used in the industry. The most common skateboard deck sizes are:

Mini deck: 7.5" - 7.75" wide

Mid-size deck: 7.75" - 8.25" wide

Full-size deck: 8.25" - 8.5" wide

Cruiser deck: 8.5" - 10" wide

It's important to note that the width of the deck is measured in inches and refers to the width of the board at its widest point. The length of the deck can also vary but is generally between 28" - 32". The size of the skateboard deck is a matter of personal preference and is influenced by the rider's style, height, and shoe size.

An average sized Skateboard Deck is selected for the implementation of path planning algorithm.

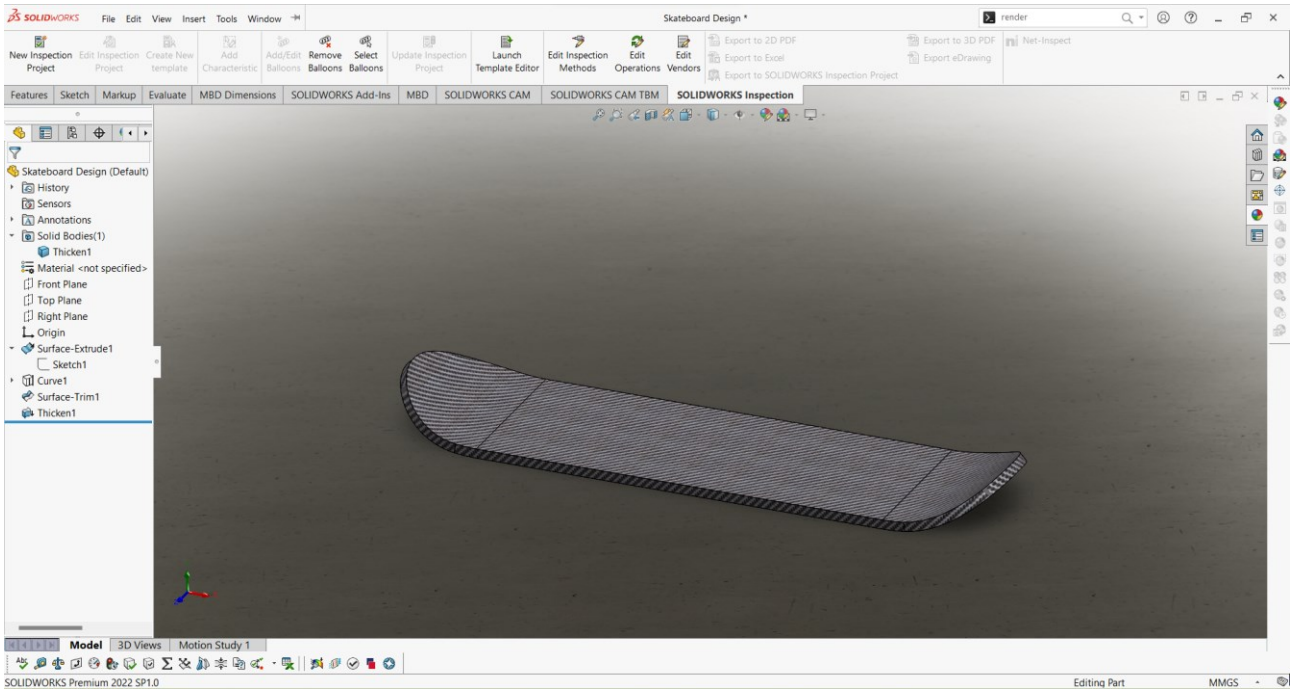


Figure 6: Skateboard 3D View

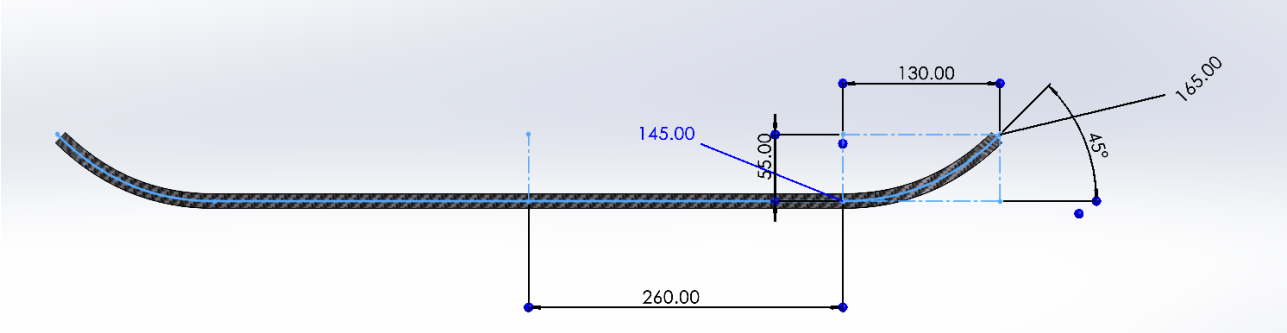


Figure 7: Skateboard Front View

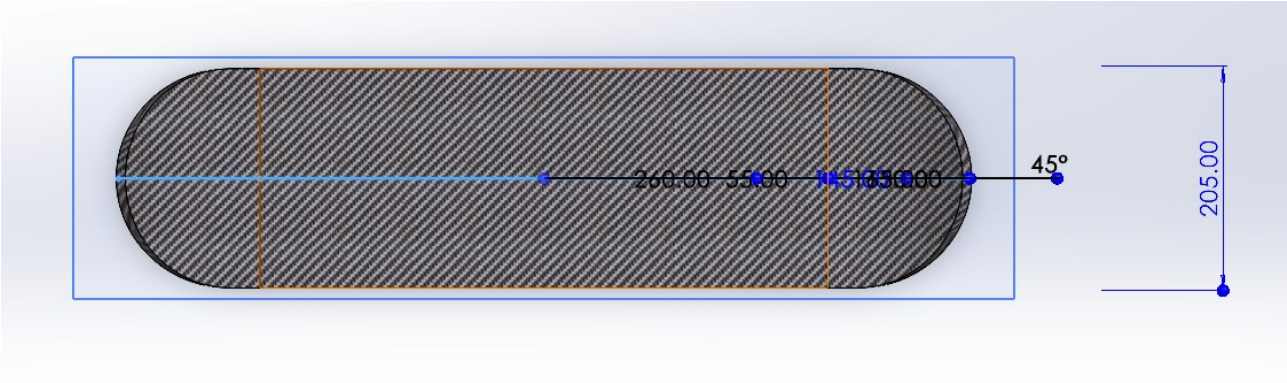


Figure 8: Skateboard Top View

Chapter 03: Implementation of Algorithm

Path planning for a UR5 robot and controlling its 6-DOF manipulator can be challenging but rewarding. To get started, you'll need to choose a suitable algorithm for path planning. Here are a few commonly used algorithms:

3.1 Factors affecting the selection of Path Planning

When selecting an algorithm for path planning, several factors need to be considered. Here are some important factors to consider:

Workspace and Environment:

Analyze the characteristics of the robot's workspace and the environment in which it operates. Consider the dimensions, shape, and complexity of the workspace, as well as the presence of obstacles, their number, size, and dynamic nature. Some algorithms are better suited for high-dimensional spaces or complex environments, while others are more efficient in simpler environments.

Robot Constraints:

Take into consideration the specific constraints and limitations of the robot, such as its kinematic structure, joint limits, velocity limits, and any other physical limitations. The algorithm should be compatible with the robot's capabilities and ensure that the generated paths are feasible and executable.

Optimality:

Determine whether finding the optimal path is critical for your application or if finding a feasible path is sufficient. Some algorithms, like A* or Dijkstra's algorithm, guarantee optimality but may be computationally expensive. If optimality is not a requirement, you can consider faster algorithms that provide feasible paths but do not guarantee the shortest or optimal solution.

Real-Time Performance:

Evaluate the time constraints for path planning. If the robot needs to plan paths in real-time or near real-time, consider algorithms that are computationally efficient and can quickly generate paths. Sampling-based algorithms like RRT or PRM are often favored for real-time applications.

Computational Resources:

Assess the available computational resources, such as processing power and memory, to determine if the algorithm can be effectively implemented within those constraints. Some algorithms require more computational resources than others, and it's important to ensure that the chosen algorithm is feasible for the hardware/software infrastructure available.

Robustness and Flexibility:

Consider the algorithm's robustness to uncertainties, noise, or changes in the environment. Some algorithms can handle dynamic environments or adapt to changing conditions more effectively. Additionally, think about the algorithm's flexibility to handle different scenarios and path planning requirements, such as multi-objective optimization, different types of motion, or additional constraints.

Implementation Complexity:

Assess the complexity and effort required to implement the algorithm. Some algorithms have well-established implementations and libraries, making it easier to integrate into existing systems. Others may require more development and customization.

By considering these factors, you can identify the most suitable algorithm that meets your specific path planning requirements and constraints. It's also important to test and evaluate different algorithms in your specific application to assess their performance and choose the best option.

3.2 Types of Algorithms:

A* (A-star) Algorithm:

A popular graph search algorithm that finds the shortest path between two points. It can be applied to plan a collision-free path for the robot.

Dijkstra's Algorithm:

Another graph search algorithm that can find the shortest path between two points. It explores the graph in a breadth-first manner and can be adapted for path planning.

Rapidly exploring Random Trees (RRT):

A sampling-based algorithm that incrementally grows a tree structure by randomly sampling the configuration space. RRT is efficient for high-dimensional spaces and can handle complex environments.

Probabilistic Roadmap (PRM):

This algorithm constructs a graph by randomly sampling the configuration space and connects the samples that are collision-free. It then finds a path between the start and goal configurations using graph search techniques.

3.3 Methodology:

Once you have chosen an algorithm, you can implement it to plan a collision-free path for the UR5 robot from its initial configuration to the desired position on the mold of the skateboard. Remember to consider the robot's kinematics and any constraints in the environment.

After planning the path, you can use inverse kinematics to determine the joint angles required for the UR5 manipulator to reach each point along the path. There are mathematical methods and libraries available for solving inverse kinematics problems.

Define the configuration space:

Start by defining the configuration space of your UR5 robot. This includes the joint limits, collision obstacles, and the start and goal configurations.

Initialize the RRT:

Create an RRT data structure to hold the nodes and edges of the tree. Initialize it with the start configuration as the root node.

Sample random configurations: Randomly sample configurations within the configuration space. Ensure that the sampled configurations are collision-free and valid for the robot.

Extend the tree:

For each sample configuration, find the nearest node in the current tree. Generate a new node by extending from the nearest node towards the sampled configuration. Ensure that the generated node is collision-free and valid.

Connect nodes:

Connect the newly generated node to the tree by adding an edge between the nearest node and the new node. This step expands the RRT tree towards the goal configuration.

Check for convergence:

Repeat steps 3 to 5 until the tree reaches the goal configuration or until a specified number of iterations.

Extract the path:

Once the tree reaches the goal configuration, trace the path from the start configuration to the goal configuration by following the edges in the tree.

Smooth the path (optional):

You can apply a path smoothing technique, such as local optimization or spline interpolation, to improve the generated path's quality and remove any unnecessary zigzagging.

Execute the trajectory:

Using inverse kinematics, calculate the joint angles required to follow the planned path. Send the joint angles to the UR5 robot controller to execute the trajectory.

In MATLAB, you can implement the RRT algorithm using custom code or leverage existing robotics libraries, such as the Robotics System Toolbox or the RVC Toolbox. These toolboxes provide built-in functions for robot modeling, kinematics, and path planning that can facilitate your implementation.

3.4 Rapidly exploring Random Trees (RRT):

Rapidly exploring Random Trees (RRT) is a popular algorithm for path planning in robotics. It offers several significant advantages, which contribute to its widespread use and significance in the field. Here are some key points about the significance of RRT:

```
RRT_VISIBILITY(q_start, q_goal)
1     T.add(q_start)
2     q_new ← q_start
3     while(!VISIBLE(q_new, q_goal))
4         q_target = RANDOM_NODE()
5         q_nearest = T.NEAREST_NEIGHBOR(q_target)
6         q_new = EXTEND(q_nearest, q_target, expansion_time)
7         if(q_new != NULL)
8             q_new.setParent(q_nearest)
9             T.add(q_new)
10    while(DISTANCE(q_new, q_goal) > d_threshold)
11        q_new = EXTEND(q_new, q_goal, expansion_time)
12        T.add(q_new)
13    ResultingPath ← T.TraceBack(q_new)
14    return ResultingPath
```

Figure 9: Key points about the significance of RRT

Benefits of RRT

Sampling-Based Approach:

RRT belongs to the family of sampling-based algorithms. It constructs a tree structure by randomly sampling configurations in the robot's configuration space. This sampling-based approach makes it well-suited for high-dimensional and complex environments where exhaustive search methods are impractical.

Probabilistic Completeness:

RRT is probabilistically complete, meaning that given enough time, it will find a solution if one exists. It is not guaranteed to find the optimal solution, but it provides a feasible solution quickly. This property makes RRT particularly useful for real-time applications where finding an approximate solution within a limited time frame is more important than finding the absolute best solution.

Exploration of Configuration Space:

RRT actively explores the configuration space by expanding the tree towards unexplored regions. This exploration behavior enables RRT to quickly cover large portions of the configuration space and find feasible paths in complex and cluttered environments.

Adaptability to Constraints:

RRT can incorporate various constraints and limitations into the path planning process. These constraints can include joint limits, collision avoidance, dynamic obstacles, or other constraints specific to the robot and environment. By considering these constraints during tree expansion, RRT can generate collision-free and valid paths.

Real-Time Path Planning:

RRT is well-suited for real-time path planning scenarios. Its efficiency and ability to incrementally grow the tree make it suitable for online planning, where the environment may change dynamically, and the robot needs to adapt its trajectory accordingly.

Versatility and Implementation Simplicity:

RRT is a versatile algorithm that can be implemented for a wide range of robot platforms, including manipulators, mobile robots, and autonomous vehicles. Its relatively simple implementation makes it accessible to researchers, students, and practitioners in the field of robotics.

These significant characteristics and advantages make RRT a valuable tool for solving path planning problems in robotics. It has been successfully applied in various domains, including industrial automation, autonomous vehicles, and robotic manipulation tasks.

It's important to note that while RRT is a powerful algorithm, there are other path planning algorithms available, each with its own strengths and limitations. The choice of algorithm depends on the specific requirements and constraints of the application at hand.

3.5 Types of RRT

RRT (Basic RRT):

The basic RRT algorithm starts with a tree consisting of a single root node and iteratively expands the tree by sampling random configurations and connecting them to the nearest node in the existing tree. The algorithm biases the sampling towards the goal region to guide the tree towards the goal configuration. This type of RRT is suitable for general path planning problems.

RRT* (RRT Star):

RRT* improves upon the basic RRT algorithm by introducing a rewiring step. After adding a new node to the tree, RRT* searches for neighbouring nodes that can potentially lead to a lower-cost path and rewires the tree to include these nodes if they improve the overall path. RRT* aims to find an optimal path with respect to a specified cost function and is particularly useful for problems where the optimality of the path is crucial.

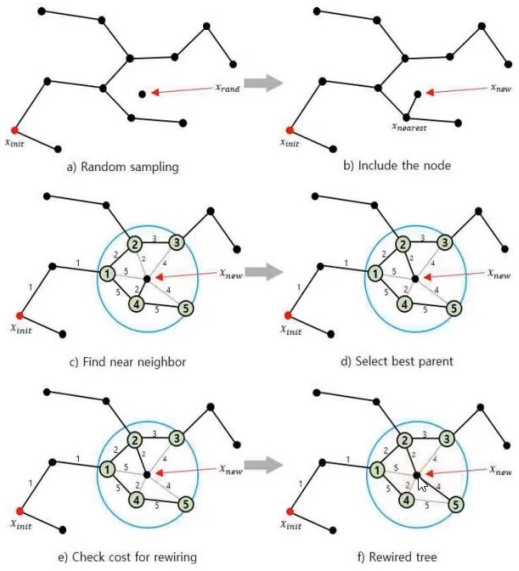


Figure 10: RRT Algorithm

RRT-Connect:

RRT-Connect is an extension of the basic RRT algorithm designed specifically for problems with two connected robots or objects. It simultaneously grows two separate RRT trees, one from the start configuration and the other from the goal configuration, until the trees connect. The algorithm alternates between growing the trees and checking for potential connections, resulting in a path connecting the start and goal configurations.

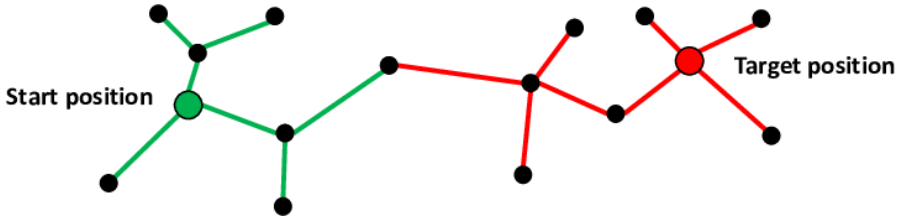


Figure 11: RRT-Connect Algorithm Description

RRT* Connect:

RRT* Connect combines the advantages of RRT* and RRT-Connect. It utilizes the rewiring technique of RRT* to improve the quality of the paths while maintaining the ability to quickly connect the start and goal configurations like RRT-Connect. RRT* Connect is suitable for problems that require both optimality and efficient connection of configurations.

Bidirectional RRT:

Bidirectional RRT simultaneously explores the configuration space from both the start and goal configurations. It grows two separate trees, one from the start and the other from the goal, and attempts to connect them by expanding towards each other. Bidirectional RRT can potentially find a solution faster by searching from both ends of the problem.

RRT_BIDIRECTIONAL(x_{init}, x_{goal})
1 $\mathcal{T}_a$.init(x_{init}); $\mathcal{T}_b$.init(x_{goal});
2 **for** $k = 1$ **to** K **do**
3 $x_{rand} \leftarrow$ RANDOM_STATE();
4 **if not** (EXTEND($\mathcal{T}_a, x_{rand}$) = *Trapped*) **then**
5 **if** (EXTEND($\mathcal{T}_b, x_{new}$) = *Reached*) **then**
6 Return PATH($\mathcal{T}_a, \mathcal{T}_b$);
7 SWAP($\mathcal{T}_a, \mathcal{T}_b$);
8 Return *Failure*

Figure 12: Bi-Directional RRT Algorithm

Chapter 04: Software Simulations and Results

4.1 Fibersim and Optimum Layup Strategy:

In order to achieve the optimum layup strategy for our project, Fibersim was utilized as a powerful software tool specifically designed for composite part design and analysis. Fibersim offers advanced capabilities for analyzing and optimizing the layup of composite materials, considering factors such as material properties, fiber orientation, and manufacturing constraints.Through the utilization of Fibersim, a comprehensive analysis of the composite layup process was conducted for our project. The software facilitated the creation of a virtual representation of the composite part, incorporating the desired fiber orientation, material properties, and manufacturing specifications. By simulating the laying process within Fibersim, we were able to evaluate the feasibility and efficiency of various layup strategies.

The analysis in Fibersim encompassed the evaluation of several factors, including fiber orientation, ply thickness, and stacking sequence. The software provided detailed insights into the structural performance of the composite part under different layup configurations. By comparing the results of different simulations, we identified the optimal layup strategy that aligned with our project's requirements in terms of strength, weight, and other performance criteria.

The determined optimum layup strategy, derived from the Fibersim analysis, was then applied during the rolling process. We relied on the results obtained from Fibersim to guide the rolling of the pre-preg layer onto the mold. By adhering to the recommended fiber orientation and ply thickness, we ensured that the composite part would exhibit the desired mechanical properties and structural integrity.

It is important to note that while the rolling process was automated, the actual laying of the layers and stacking was still performed manually during this phase of the project. The ultimate objective of our project is to fully automate the layup process, utilizing three collaborative robots. However, we made significant progress by successfully automating the rolling process, which served as a crucial step towards further automation advancements.

In conclusion, the utilization of Fibersim and the determination of the optimum layup strategy played a pivotal role in the success of our project. The software provided valuable insights into the performance of the composite part and empowered us to make well-informed decisions throughout the layup process. By integrating the results from Fibersim into the rolling process, we ensured the manufacture of the composite part with the optimal fiber orientation and ply thickness, paving the way for future automation advancements within our project.

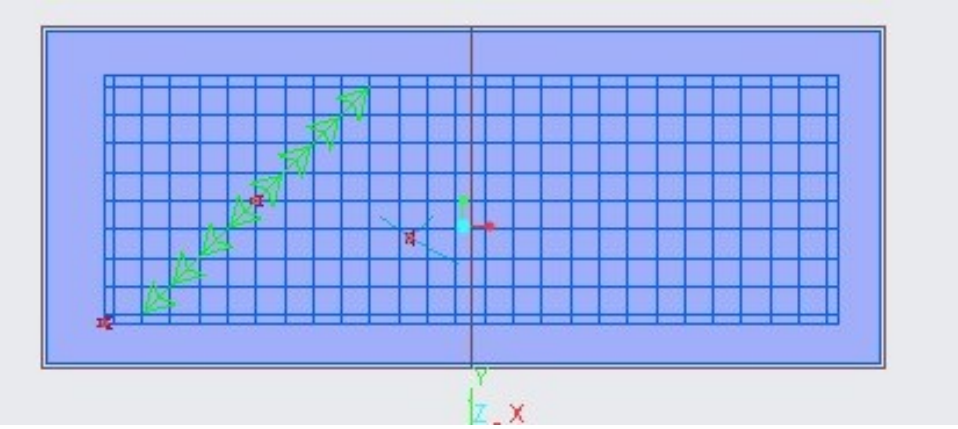


Figure 13 Fibersim path feasibility

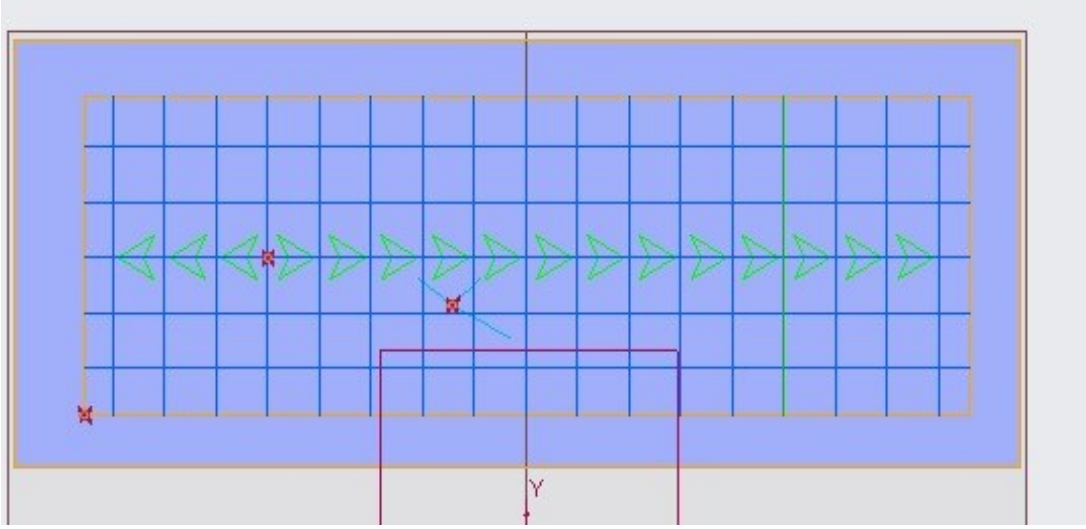


Figure 14 Fibersim Path feasibility

4.2 Implementation of ROS, MoveIt, and RQT for Path Planning Evaluation:

In order to evaluate different algorithms for optimum path planning, the project utilized ROS (Robot Operating System), MoveIt, and RQT. These software frameworks provided the necessary tools and capabilities to simulate, analyze, and integrate the path planning algorithms into the project's robotic system.

ROS, an open-source middleware, served as the foundation for the project's software development. It facilitated the communication and coordination between the different components of the system, including the robots, sensors, and software modules. By utilizing ROS, the project achieved modularity and flexibility in its software architecture, enabling seamless integration of various functionalities.

MoveIt, a widely used motion planning framework for ROS, played a crucial role in the project. It provided high-level interfaces and tools for motion planning, kinematics, collision checking, and trajectory execution. The project leveraged MoveIt to implement and evaluate the performance of

different path planning algorithms, namely A* (A-star), RRT (Rapidly-exploring Random Tree), and Dijkstra. These algorithms were chosen based on their suitability for the project's requirements and the potential to provide optimized paths for the robots.

The project extensively conducted software simulations using ROS and MoveIt to analyze the feasibility and efficiency of the automated layup application. Through these simulations, the project could evaluate the performance of the chosen path planning algorithms under different scenarios and environments. The simulations allowed for testing and refining the algorithms, ensuring they met the project's objectives and requirements.

Additionally, rqt, a graphical user interface (GUI) framework for ROS, was utilized in the project to provide a user-friendly interface for monitoring and controlling the robotic system. It allowed the project team to visualize and interact with the robot's motions, trajectories, and path planning algorithms. The integration of rqt into the project's software architecture enabled convenient monitoring and debugging of the system during both simulations and real-world execution.

The project's evaluation of the path planning algorithms involved comparing their performance based on factors such as planning time, collision avoidance, and overall trajectory optimization. The simulation results provided valuable insights into the strengths and weaknesses of each algorithm, aiding in the selection and optimization process. Furthermore, the project emphasized the integration of the chosen algorithms into the real-world hardware, specifically the UR5 manipulators. This integration phase involved evaluating the robots, optimizing the algorithms, and ensuring seamless communication between the robot controller and the system software. Rigorous testing and iterations were conducted to address any disparities between the simulation results and the execution in the real-world environment.

Overall, the utilization of ROS, MoveIt, and rqt in the project allowed for the evaluation, optimization, and integration of different path planning algorithms. These software frameworks provided the necessary tools and functionalities to simulate, analyze, and control the robotic system, contributing to the project's objective of automating the layup process.

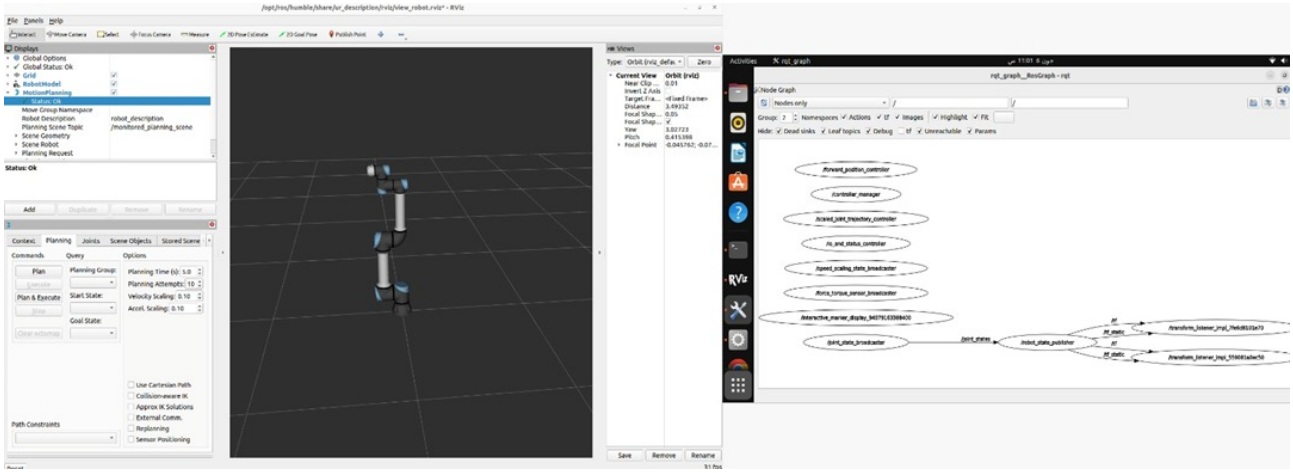


Figure 15 Integrating UR5 with MoveIt

4.3 Integration of Path Planning Algorithms into UR5 Manipulators for Real-World Hardware:

During the integration phase, our project focused on seamlessly incorporating the path planning algorithms into the real-world hardware, specifically the UR5 manipulators. We recognized the importance of evaluating the robots themselves to ensure they could effectively execute the planned trajectories. This evaluation process involved assessing their mechanical capabilities, such as their range of motion, speed, and payload capacity, to align them with the requirements of the automated layup application. Simultaneously, we optimized the path planning algorithms to enhance their performance in the context of the UR5 manipulators. By fine-tuning the algorithms, we aimed to achieve more efficient and accurate motion planning, minimizing unnecessary movements and maximizing the robots' productivity during the layup process.

One critical aspect of successful integration was establishing smooth communication between the robot controller and the system software, namely RobotDK and ROS (Robot Operating System). This connection enabled seamless data exchange, allowing the robot controller to receive the planned trajectories generated by the path planning algorithms and execute them accurately. We paid careful attention to ensuring compatibility and reliability between the different software components, implementing robust communication protocols to mitigate any potential issues.

To validate the integration and guarantee the algorithms' effectiveness in the real-world environment, extensive testing and iterations were conducted. We placed great emphasis on addressing any discrepancies that arose between the simulations and the actual execution. These tests allowed us to fine-tune parameters, calibrate the robots' movements, and identify potential areas for improvement. By iteratively refining the system based on these tests, we aimed to achieve a high level of consistency and accuracy in the layup process. Throughout this integration phase, our project team worked diligently to bridge the gap between theoretical simulations and practical implementation. We recognized the complexity of translating algorithmic plans into physical actions and strived to create a reliable and efficient system that aligned with our project objectives. The integration process served as a crucial stepping stone, paving the way for the ultimate goal of fully automating the layup process using three collaborative robots.

In conclusion, the project's integration phase involved evaluating the UR5 manipulators, optimizing the path planning algorithms, and ensuring seamless communication between the robot controller and the system software. Rigorous testing and iterations were performed to address any discrepancies between simulations and real-world execution, thereby enhancing the project's understanding and control over the layup process.

Figure 16 Real time MoveIt simulation

4.4 Inverse Kinematics Solutions for 6 DOF (RoboAnalyzer):

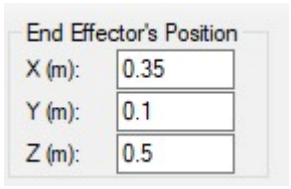


Figure 17: RoboAnalyzer End Effector Position

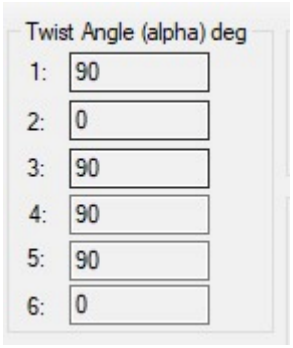


Figure 18 RoboAnalyzer Twist Angles

4.5 UR5 Inverse Kinematics Results:

Following are the 8 possible combinations of angles provided by inverse kinematics. All these solutions can be used in forward kinematics to generate the same path.

For point 1

End-effector Position (-0.25, 0.403, 0.08)

```
Joint Angle Solutions
Solution 1 -> 58.19 36.92 46.88 0.00 90.00 90.00

Solution 2 -> 58.19 5.63 -15.78 0.00 90.00 90.00

Solution 3 -> 58.19 -24.02 46.88 0.00 90.00 90.00

Solution 4 -> 58.19 5.40 -15.78 0.00 90.00 90.00

Solution 5 -> -58.19 58.08 86.13 -0.00 90.00 90.00

Solution 6 -> -58.19 25.16 -55.03 -0.00 90.00 90.00

Solution 7 -> -58.19 -40.58 86.13 -0.00 90.00 90.00

Solution 8 -> -58.19 -13.44 -55.03 -0.00 90.00 90.00
```

Figure 19: Inverse Kinematics Results for Point 1

For point 2

End-effector Position (0.25, 0.403, 0.08)

```
Joint Angle Solutions
Solution 1 -> 58.19 58.08 86.13 0.00 90.00 90.00

Solution 2 -> 58.19 25.16 -55.03 0.00 90.00 90.00

Solution 3 -> 58.19 -40.58 86.13 0.00 90.00 90.00

Solution 4 -> 58.19 -13.44 -55.03 0.00 90.00 90.00

Solution 5 -> -58.19 36.92 46.88 -0.00 90.00 90.00

Solution 6 -> -58.19 5.63 -15.78 -0.00 90.00 90.00

Solution 7 -> -58.19 -24.02 46.88 -0.00 90.00 90.00

Solution 8 -> -58.19 5.40 -15.78 -0.00 90.00 90.00
```

Figure 20: Inverse Kinematics Results for Point 2

For point 3

End-effector Position (0.25, 0.5, 0.08)

```
Joint Angle Solutions
Solution 1 -> 63.43 54.09 79.01 0.00 90.00 90.00

Solution 2 -> 63.43 32.74 -69.89 0.00 90.00 90.00

Solution 3 -> 63.43 -37.80 79.01 0.00 90.00 90.00

Solution 4 -> 63.43 -20.34 -69.89 0.00 90.00 90.00

Solution 5 -> -63.43 46.18 64.45 -0.00 90.00 90.00

Solution 6 -> -63.43 14.29 -33.35 -0.00 90.00 90.00

Solution 7 -> -63.43 -31.76 64.45 -0.00 90.00 90.00

Solution 8 -> -63.43 -3.12 -33.35 -0.00 90.00 90.00
```

Figure 21: Inverse Kinematics Results for Point 3

For point 4

End-effector Position (-0.25, 0.5, 0.08)

```
Joint Angle Solutions
Solution 1 -> 63.43 46.18 64.45 0.00 90.00 90.00

Solution 2 -> 63.43 14.29 -33.35 0.00 90.00 90.00

Solution 3 -> 63.43 -31.76 64.45 0.00 90.00 90.00

Solution 4 -> 63.43 -3.12 -33.35 0.00 90.00 90.00

Solution 5 -> -63.43 54.09 79.01 -0.00 90.00 90.00

Solution 6 -> -63.43 32.74 -69.89 -0.00 90.00 90.00

Solution 7 -> -63.43 -37.80 79.01 -0.00 90.00 90.00

Solution 8 -> -63.43 -20.34 -69.89 -0.00 90.00 90.00
```

Figure 22: Inverse Kinematics Results for Point 4

For point 5

End-effector Position (-0.25, 0.57, 0.08)

```
Joint Angle Solutions
Solution 1 -> 66.32 53.31 77.59 0.00 90.00 90.00

Solution 2 -> 66.32 20.85 -46.49 0.00 90.00 90.00

Solution 3 -> 66.32 -37.24 77.59 0.00 90.00 90.00

Solution 4 -> 66.32 -9.40 -46.49 0.00 90.00 90.00

Solution 5 -> -66.32 47.32 66.58 -0.00 90.00 90.00

Solution 6 -> -66.32 39.19 -82.32 -0.00 90.00 90.00

Solution 7 -> -66.32 -32.67 66.58 -0.00 90.00 90.00

Solution 8 -> -66.32 -25.97 -82.32 -0.00 90.00 90.00
```

Figure 23: Inverse Kinematics Results for Point 5

For point 6

End-effector Position (0.25, 0.57, 0.08)

```
Joint Angle Solutions
Solution 1 -> 66.32 47.32 66.58 0.00 90.00 90.00

Solution 2 -> 66.32 39.19 -82.32 0.00 90.00 90.00

Solution 3 -> 66.32 -32.67 66.58 0.00 90.00 90.00

Solution 4 -> 66.32 -25.97 -82.32 0.00 90.00 90.00

Solution 5 -> -66.32 53.31 77.59 -0.00 90.00 90.00

Solution 6 -> -66.32 20.85 -46.49 -0.00 90.00 90.00

Solution 7 -> -66.32 -37.24 77.59 -0.00 90.00 90.00

Solution 8 -> -66.32 -9.40 -46.49 -0.00 90.00 90.00
```

Figure 24: Inverse Kinematics Results for Point

For point 7

End-effector Position (-0.35, 0.5, 0.00)

```
Joint Angle Solutions
Solution 1 -> 55.01 29.33 44.53 0.00 90.00 90.00

Solution 2 -> 55.01 1.04 -13.43 0.00 90.00 90.00

Solution 3 -> 55.01 -29.33 44.53 0.00 90.00 90.00

Solution 4 -> 55.01 -1.04 -13.43 0.00 90.00 90.00

Solution 5 -> -55.01 41.70 70.12 -0.00 90.00 90.00

Solution 6 -> -55.01 30.86 -78.78 -0.00 90.00 90.00

Solution 7 -> -55.01 -41.70 70.12 -0.00 90.00 90.00

Solution 8 -> -55.01 -30.86 -78.78 -0.00 90.00 90.00

>>
```

Figure 25: Inverse Kinematics Results for Point 7

For point 8

End-effector Position (0.28, 0.405, 0.066)

```
Joint Angle Solutions
Solution 1 -> 55.34 58.03 88.82 0.00 90.00 90.00

Solution 2 -> 55.34 25.49 -57.72 0.00 90.00 90.00

Solution 3 -> 55.34 -43.17 88.82 0.00 90.00 90.00

Solution 4 -> 55.34 -15.74 -57.72 0.00 90.00 90.00

Solution 5 -> -55.34 32.92 41.27 -0.00 90.00 90.00

Solution 6 -> -55.34 7.19 -10.17 -0.00 90.00 90.00

Solution 7 -> -55.34 -22.57 41.27 -0.00 90.00 90.00

Solution 8 -> -55.34 1.92 -10.17 -0.00 90.00 90.00
```

Figure 26: Inverse Kinematics Results for Point 8

For point 9

End-effector Position (0.28, 0.55, 0.066)

```
Joint Angle Solutions
Solution 1 -> 63.02 46.78 68.00 0.00 90.00 90.00

Solution 2 -> 63.02 37.29 -80.90 0.00 90.00 90.00

Solution 3 -> 63.02 -34.58 68.00 0.00 90.00 90.00

Solution 4 -> 63.02 -26.48 -80.90 0.00 90.00 90.00

Solution 5 -> -63.02 46.86 68.15 -0.00 90.00 90.00

Solution 6 -> -63.02 15.15 -37.05 -0.00 90.00 90.00

Solution 7 -> -63.02 -34.64 68.15 -0.00 90.00 90.00

Solution 8 -> -63.02 -5.89 -37.05 -0.00 90.00 90.00
```

Figure 27: Inverse Kinematics Results for Point 9

For point 10

End-effector Position (0.31, 0.52, 0.03)

```
Joint Angle Solutions
Solution 1 -> 59.20 44.94 70.97 0.00 90.00 90.00

Solution 2 -> 59.20 32.86 -77.93 0.00 90.00 90.00

Solution 3 -> 59.20 -39.27 70.97 0.00 90.00 90.00

Solution 4 -> 59.20 -28.03 -77.93 0.00 90.00 90.00

Solution 5 -> -59.20 37.53 56.19 -0.00 90.00 90.00

Solution 6 -> -59.20 6.74 -25.09 -0.00 90.00 90.00

Solution 7 -> -59.20 -32.43 56.19 -0.00 90.00 90.00

Solution 8 -> -59.20 -2.59 -25.09 -0.00 90.00 90.00
```

Figure 28: Inverse Kinematics Results for Point 10

For point 11

End-effector Position (0.31, 0.48, 0.03)

```
Joint Angle Solutions
Solution 1 -> 57.14 48.28 77.60 0.00 90.00 90.00

Solution 2 -> 57.14 29.56 -71.30 0.00 90.00 90.00

Solution 3 -> 57.14 -42.27 77.60 0.00 90.00 90.00

Solution 4 -> 57.14 -24.89 -71.30 0.00 90.00 90.00

Solution 5 -> -57.14 33.72 48.55 -0.00 90.00 90.00

Solution 6 -> -57.14 2.99 -17.45 -0.00 90.00 90.00

Solution 7 -> -57.14 -28.84 48.55 -0.00 90.00 90.00

Solution 8 -> -57.14 1.14 -17.45 -0.00 90.00 90.00
```

Figure 29: Inverse Kinematics Results for Point 11

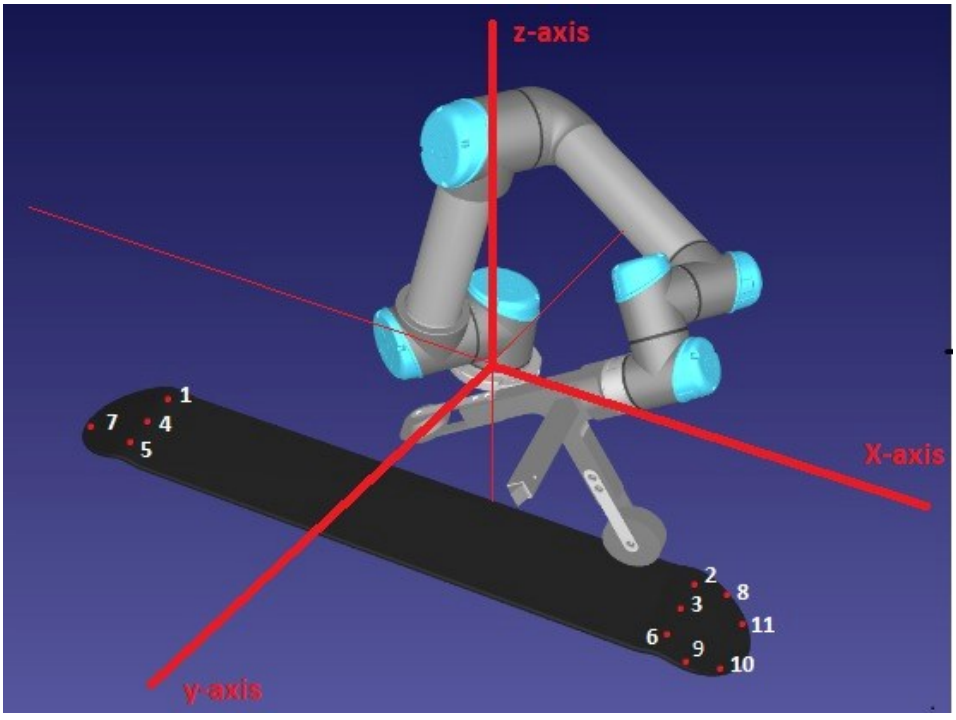


Figure 30: RoboDK Simulation of UR

Chapter 05: Fabrication

5.1 Materials Selection

Properties of Glass fiber

Tensile Strength:

Typical tensile strength: 2000-4000 MPa (290,000-580,000 psi) [7]

Specific tensile strength: 3.5-6.0 MPa·m^3/kg [8]

Density:

Typical density: 2.54-2.70 g/cm^3 (158-168 lb/ft^3) [9]

Flexural Strength:

Typical flexural strength: 300-700 MPa (44,000-101,000 psi) [10]

Chemical Resistance:

Fiberglass exhibits excellent resistance to various chemicals, including acids, alkalis, and solvents. The specific resistance may vary depending on the type of resin used in the composite material.

Thermal Resistance:

Thermal conductivity: 0.035-0.045 W/(m·K) [11]

Heat deflection temperature: 180-250°C (356-482°F)

[12]

Electrical Insulation:

Volume resistivity: 10^12-10^16 Ω·cm

[13]

Why choose glass fiber over Carbon fiber?

Strength and Stiffness:

Carbon fiber:

Carbon fiber has a higher specific strength and stiffness compared to glass fiber. It offers an excellent strength-to-weight ratio, making it suitable for lightweight and high-performance applications.

Glass fiber:

Glass fiber has lower specific strength and stiffness compared to carbon fiber. While it is not as strong or stiff, it still provides good strength and stiffness properties for many applications.

Cost:

Glass fiber:

Glass fiber is generally more cost-effective compared to carbon fiber. It is widely available and less expensive to produce, making it a more economical choice for many applications.

Carbon fiber:

Carbon fiber is relatively expensive compared to glass fiber. The production process and raw material costs contribute to its higher price point.

Weight:

Carbon fiber:

Carbon fiber is significantly lighter than glass fiber, which contributes to weight reduction in the final printed parts. This is advantageous in applications where lightweight components are desired, such as aerospace and automotive industries.

Glass fiber:

Glass fiber is heavier than carbon fiber, but it still offers weight savings compared to other traditional materials like metals.

Mechanical Properties:

Carbon fiber:

Carbon fiber exhibits exceptional mechanical properties, including high tensile strength, stiffness, and fatigue resistance. It is suitable for applications requiring superior mechanical performance.

Glass fiber:

Glass fiber has good mechanical properties, including moderate tensile strength and stiffness. It provides sufficient strength for many applications but may not offer the same performance as carbon fiber in demanding scenarios.

Thermal Conductivity:

Carbon fiber:

Carbon fiber has low thermal conductivity, making it suitable for applications requiring thermal insulation or conductivity control.

Glass fiber:

Glass fiber has higher thermal conductivity compared to carbon fiber. It may not provide the same level of thermal insulation as carbon fiber.

Electrical Conductivity:

Carbon fiber:

Carbon fiber exhibits excellent electrical conductivity, making it suitable for applications requiring electrical grounding or electromagnetic shielding.

Glass fiber:

Glass fiber is electrically insulating, which can be advantageous in applications where electrical insulation is necessary.

Ultimately, the choice between glass fiber and carbon fiber for additive manufacturing depends on specific requirements, performance needs, budget constraints, and weight considerations of the intended application.

Products manufactured by glass fiber:

Glass fiber cloth, also known as fiberglass cloth or fiberglass fabric, is used in various industries for its strength, durability, and resistance to heat and chemicals. It is primarily used as a reinforcement material in composites and is commonly found in the following products:

Fiberglass Reinforced Plastic (FRP) products:

Fiberglass cloth is used to reinforce and provide strength to FRP products such as tanks, pipes, panels, and structural components.

Boats and marine products:

Fiberglass cloth is widely used in the construction of boats, yachts, and other marine vessels. It provides strength, rigidity, and water resistance.

Aerospace components:

Glass fiber cloth is used in the manufacturing of aerospace components such as aircraft fuselages, wings, and interior panels. It offers lightweight strength and resistance to high temperatures.

Automotive parts:

Fiberglass cloth is utilized in the production of various automotive parts, including body panels, hoods, spoilers, and interior components.

Sporting goods:

Many sporting goods, such as surfboards, paddleboards, kayaks, and hockey sticks, are manufactured using fiberglass cloth for its lightweight and strong properties.

Wind turbine blades:

The blades of wind turbines are often constructed with fiberglass cloth to provide strength and durability while maintaining aerodynamic efficiency.

Construction materials:

Fiberglass cloth is used in the production of roofing materials, insulation, reinforcement for concrete structures, and corrosion-resistant linings for tanks and pipes.

Electrical insulation:

Glass fiber cloth is employed in electrical applications as insulation for wires, cables, and circuit boards due to its high electrical resistance and fire-retardant properties.

These are just a few examples of the products produced using glass fiber cloth. Its versatility and desirable properties make it a valuable material in various industries.

5.2 Preparation of Prepreg

In the preparation of prepreg, Resin Matrix (Epoxy) was first produced by mixing ***2 Huntsman Araldite LY 5052 Epoxy Resin (368 gm)*** with epoxy hardener in blender according to the following calculations:

GSM = 400

Area of skateboard = A = 72 cm x 20 cm = 0.144 m^2

30% extra area A = 0.0368 m^2 (1 ply)

Area x GSM = 14.72 grams (per ply)

For 25 plies = 14.72 x 25 = 368 grams of resin

The epoxy was mixed with hardener in 100:2 .i.e.,10 gm of hardener was used for 368 grams of epoxy.

The epoxy was manually applied on glass fiber sheets (***EGP200 E-Glass Fabric, 200g/m2, Plain weave***.)
The prepreg was left to partially cure in a self-made oven for 1 Day.

Figure 31: Mixing of Epoxy Resin

Figure 32: Prepreg sheets ready for layup

5.3 Automated rolling in Layup of Prepreg:

Automated rolling of Prepreg sheets was done using UR5 in National Centre for Robotics and Automation (NCRA).Two collaborative robot will be picking and placing the prepreg sheets as the pre layup. Picking and placing to stack up will be done in the second year of this project.

The skateboard was used as a mold and was placed in a fixed position. End effector was fixed on the robot.

By using RRT as the path planning algorithm, prepreg sheets were placed one by one and were layed up by the robotic arm.

Figure 33: Automated Layup using UR5.

5.4 Curing Process:

Prepreg sheets are partially cured but they need curing after the layup process. For this purpose, a self-made oven was used. The curing took 2-3 days because of improper temperatures.

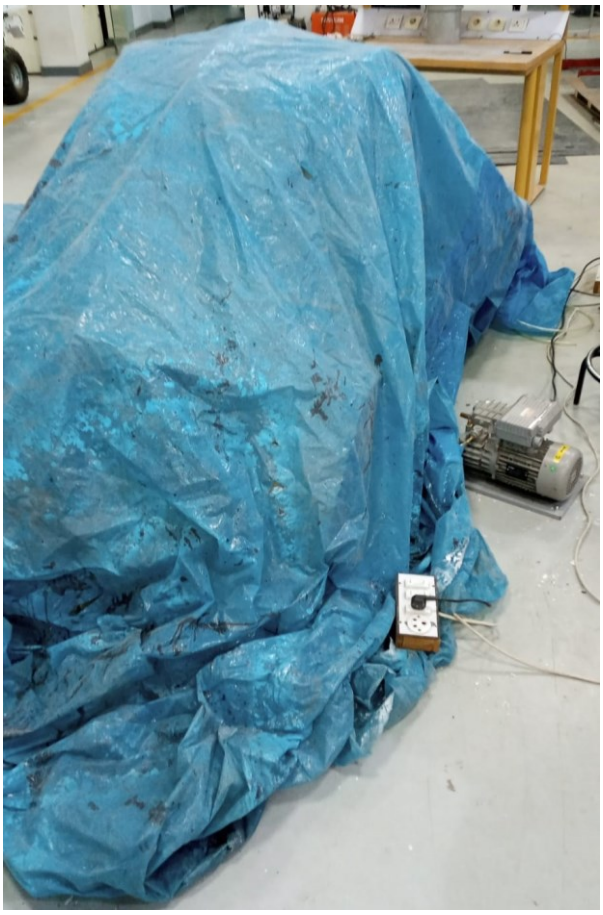


Figure 34: Oven arrangements for curing of prepreg

REFERENCES

[1] What are composites https://discovercomposites.com/what-are-composites/

[2] Automated Fiber Placement Path Planning: A state-of-the-art review Rousseau G, Wehbe R, Halbritter J et al. 2019

[3] LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical Report, Computer Science Department, Iowa State University.

[4] Liu, L., Wang, L., Chen, J., & Li, Z. (2015). Path Planning for Automated Composite Panel Layup with a Robot Manipulator Based on Potential Fields and B-spline Curves. International Journal of Advanced Robotic Systems, 12(4), 54.

[5] Gabriele Porcelli: Dynamic parameters identification of a UR5 robot manipulator. Rel. Massimo Sorli, Andrea Raviola, Stefano Paolo Pastorelli, Stefano Mauro. Politecnico di Torino, Corso di laurea magistrale in Ingegneria Meccanica (Mechanical Engineering), 2020

[6] Skateboard deck materials selection Haoyu Liu, Tasha Coote, Aiolos,Charlie Material Engineering of The University of British Columbia,Vancouver,Canada

[7] Composite Materials: Science and Engineering by Krishan K. Chawla

[8] Fiberglass and Glass Technology: Energy-Friendly Compositions and Applications by L.F. Cabeza, J. M. Gómez-Villalba, and R. Fort

[9] Fiberglass and Glass Technology: Energy-Friendly Compositions and Applications by L.F. Cabeza, J. M. Gómez-Villalba, and R. Fort

[10] Composite Materials: Science and Engineering by Krishan K. Chawla

[11] Composite Materials: Science and Engineering by Krishan K. Chawla

[12] Composite Materials: Science and Engineering by Krishan K. Chawla

[13] Reference: Fiberglass and Glass Technology: Energy-Friendly Compositions and Applications by L.F. Cabeza, J. M. Gómez-Villalba, and R. Fort

APPENDIX I

MATLAB Code for UR5 Simulation:

```
clc
clear all
format long
Pos_X = -0.25;
Pos_Y =0.402;
Pos_Z =  0.08;
ox = 0;
oy = 0;
oz = 0;
ax = 0;
ay = 0;
az = 0;
% DH Parameters
r2 = 0.425;
r3 = 0.39225;
d1 = 0.08916;
d2=0;
d3=0;
d4=0.10915;
d5=0.9456;
d6=0.0823;


Soln_1 = inverse_kinematics(1,1,1,Pos_X,Pos_Y, Pos_Z, ox, oy, oz, ax, ay, az, r2, r3, d3, d4);
Soln_2 = inverse_kinematics(1,-1,1,Pos_X,Pos_Y, Pos_Z, ox, oy, oz, ax, ay, az, r2, r3, d3, d4);
Soln_3 = inverse_kinematics(1,1,-1,Pos_X,Pos_Y, Pos_Z, ox, oy, oz, ax, ay, az, r2, r3, d3, d4);
Soln_4 = inverse_kinematics(1,-1,-1,Pos_X,Pos_Y, Pos_Z, ox, oy, oz, ax, ay, az, r2, r3, d3, d4);
Soln_5 = inverse_kinematics(-1,1,1,Pos_X,Pos_Y, Pos_Z, ox, oy, oz, ax, ay, az, r2, r3, d3, d4);
Soln_6 = inverse_kinematics(-1,-1,1,Pos_X,Pos_Y, Pos_Z, ox, oy, oz, ax, ay, az, r2, r3, d3, d4);
Soln_7 = inverse_kinematics(-1,1,-1,Pos_X,Pos_Y, Pos_Z, ox, oy, oz, ax, ay, az, r2, r3, d3, d4);
Soln_8 = inverse_kinematics(-1,-1,-1,Pos_X,Pos_Y, Pos_Z, ox, oy, oz, ax, ay, az, r2, r3, d3, d4);
```

```
Soln = [Soln_1;Soln_2;Soln_3;Soln_4;Soln_5;Soln_6;Soln_7;Soln_8];
fprintf("Joint Angle Solutions\n");
for i = 1:8
   fprintf("Solution %d -> ",i);
   for j = 1:6
      fprintf("%.2f ",Soln(i,j));
   end
   fprintf("\n\n");
end


function Joint_Angle = inverse_kinematics(Number_1,Number_2,Number_3,Pos_X,Pos_Y, Pos_Z, ox, oy, oz, ax, ay, az, r2, r3, d3, d4)
D1 = sqrt(4*d3^2*Pos_X^2-4*(Pos_X^2+Pos_Y^2)*(-Pos_Y^2+d3^2));
A = (2*d3*Pos_X + Number_1*D1)/(2*(Pos_X^2+Pos_Y^2));
Theta_1 = asind(A);
Theta_1;
K3 = cosd(Theta_1)^2*Pos_X^2+sind(Theta_1)^2*Pos_Y^2+Pos_Z^2+2*cosd(Theta_1)*sind(Theta_1)*Pos_X*Pos_Y-r2^2-r3^2-d4^2;
D3 = sqrt((4*K3*r2*d4)^2+4*(4*r2^2*r3^2-K3^2)*(4*r2^2*d4^2+4*r2^2*r3^2));
C = (-4*K3*r2*d4 + Number_2*D3)/(2*(4*r2^2*d4^2+4*r2^2*r3^2));
Theta_3 = asind(C);
Theta_3;
D2 = sqrt((2*Pos_Z*(sind(Theta_3)*d4-cosd(Theta_3)*r3-r2))^2 - 4 * (Pos_Z^2-(cosd(Theta_3)*d4+sind(Theta_3)*r3)^2) * ((sind(Theta_3)*d4-cosd(Theta_3)*r3-r2)^2+(cosd(Theta_3)*d4+sind(Theta_3)*r3)^2));
B = (-1*(2*Pos_Z*(sind(Theta_3)d4-cosd(Theta_3)*r3-r2)) + Number_3*D2)/(2((sind(Theta_3)*d4-cosd(Theta_3)*r3-r2)^2+(cosd(Theta_3)*d4+sind(Theta_3)*r3)^2));
Theta_2 = asind(B);
Theta_2;
Theta_23 = Theta_2 + Theta_3;
Theta_5 = acosd(-cosd(Theta_1)*sind(Theta_23)*ax-sind(Theta_1)*sind(Theta_23)*ay+cosd(Theta_23)*az);
```

```
Theta_4 = asind((sind(Theta_1)*ax - cosd(Theta_1)*ay)/sind(Theta_5));

Theta_6 = acosd(-1*(ox*(cosd(Theta_1)cosd(Theta_23)*sind(Theta_4) + cosd(Theta_4)*sind(Theta_1)) - oy(cosd(Theta_1)*cosd(Theta_4) - sind(Theta_1)*sind(Theta_4)*cosd(Theta_23)) + oz*sind(Theta_4)*sind(Theta_23)));

Joint_Angle = [Theta_1,Theta_2,Theta_3,Theta_4,Theta_5,Theta_6];

end
```