HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

GRADUATION THESIS

The development of human-collision avoidance behavior for robotic manipulator by combine RRT and adaptive potential field algorithm.

Trinh Dam Nhan
nhan.td195802@sis.hust.edu.vn

Advanced Mechatronics Program

Supervisor: Associate Professor Bùi Hải Lê _____
Signature

Senior Architect Hoàng Thanh Thuỷ _____
Signature

Department: Department of Mechatronics

School: School of Mechanical Engineering

Acknowledgement

In an era where industrial production is continuously evolving and advancing, industrial robots have become a crucial pillar, opening up promising prospects for optimizing production processes and enhancing efficiency. Coupled with the powerful wave of Industry 4.0 technology, industrial robots have been able to integrate artificial intelligence along with new controllers to become smarter, more versatile, and nearly capable of completely replacing humans in tedious, difficult, and dangerous tasks.

With the desire to learn, research, and apply the knowledge acquired in the classroom to this field, specifically focusing on planning systems and vision systems in robots, we chose the topic: **"The development of human-collision avoidance behavior for robotic manipulator by combine RRT and adaptive potential field algorithm."**.

During the implementation of the thesis, we faced many challenges and difficulties. However, thanks to the dedicated guidance of Associate Professor Dr. Mạc Thị Thoa, from the School of Mechanical Engineering - Hanoi University of Science and Technology, along with the enthusiastic support from lab members, our team was able to meet the set requirements.

Despite our best efforts, this graduation thesis inevitably contains certain shortcomings and limitations. Therefore, we sincerely look forward to receiving feedback and suggestions from the teachers to optimize and perfect the topic.

Finally, I would like to express my gratitude to Ms. Mạc Thị Thoa for her guidance, advice, and provision of resources over the past few months, which have facilitated the completion of this thesis.

Project Overview

In this graduation thesis, our biggest goal is designing a path planner, control a 6 DOF robot manipulator to execute the path in a human-shared workspace, meanwhile detecting static and dynamic obstacle to safely replan the path when encounter human's hand.

To achieve this, we decided to research and ultilize controllers, planners and camera modules to detecting human, plannning the path and controlling the robot. These can be done with using Robotic Operating System - ROS to simulate and experiment. We also have done extensive research on different path planners, such as RRT, RRT*, AAPF, A*.. and propose a hybrid planner.

For the RRT and AAPF planners, we investigated, studied, and applied them to create the robot's path. We then evaluated the advantages and disadvantages of the two planners.

From there, we proposed a new hybrid planner combining both methods, with RRT used as the global planner and AAPF as the local planner. In our experiments, the robot's path was executed in a map with known obstacles (open box) and potential dynamic obstacles (human arm). RRT was used to avoid the box and find a feasible path. These sets of points formed the *reference path*. Subsequently, AAPF was used to plan the segments between consecutive trajectories in the set, continuously updating the *planning scene* to calculate repulsive and attractive forces when the human arm appeared.

We also developed a vision module for the robot using a 3D camera, Aruco Marker, and applied artificial intelligence models to determine the position and coordinates of obstacles. For the robot to avoid obstacles, specifically humans, it is necessary to identify the position and depth of human body parts in the image to plan the path accurately.

Finally, we simulated the trajectory in Rviz and Gazebo, conducting multiple tests before controlling the actual robot. When operating the real robot, we were very careful and managed to fully control the robot, completing the project without any component damage.

Student
Signature

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.1 Industrial Robot Manipulator

Robotics is a relatively new field of study compared to many other academic disciplines, with the ultimate, albeit somewhat ambiguous, goal of creating a mechanical structure that thinks and acts like a human. The effort to create a machine with human-like intelligence prompts us to question our own bodies: why is the human body structured in such a way that it allows us to grasp objects flexibly or perform complex movements with ease? These are actions we have been doing unconsciously for ages, but they become remarkably complex when we attempt to understand them. Due to this objective, robotics consistently captivates students and researchers with the curiosity and intrigue it inspires.

Manipulators is a classical research field in Robotics and Mechatronics, playing an irreplaceable role in modern industries. Designed to perform repetitive tasks with high precision and inherent danger, manipulators are utilized in numerous automated manufacturing processes, from heavy industries such as automobile production to advanced industries like semiconductor chip manufacturing.

Taking inspiration from nature, the manipulator is designed based on the human arm, with the purposed of mimicking the tasks as grasping, holding and picking. The first robotic arm, named "Unimate," created by General Motors in the 1960s, marked a significant breakthrough, allowing the automation of assembly processes and reducing the heavy and dangerous tasks typically encountered by workers on the production line. Since then, robotic arms have undergone numerous designs and are manufactured in various models to serve specific purposes, offering high flexibility, great precision, and the capability to perform complex tasks.

Figure 1.1: Unimate - the first industrial manipulator

Industrial manufacturing is the field where robotic arms are most widely applied. New systems are designed with multiple joints and equipped with force sensors and cameras, combined with the exceptional advancements in computing, enabling robotic lines to undertake complex tasks with high precision. Robotic arms can replace workers on welding lines, welding curves with millimeter accuracy and capable of welding at any angle within

the work environment. Robotic arms equipped with cameras and artificial intelligence can also sort goods in logistics lines, a hot topic that major tech companies like Amazon, Shopee, and Viettel are applying and continuously researching and developing.

Figure 1.2: Wielding manipulator

Figure 1.3: Amazon's manipulator sorting packages

Additionally, the development of the flexibility and intelligence of robotic arms opens up their potential in fields such as healthcare, food processing, and the semiconductor industry. The most notable example is the Da Vinci surgical robot system, which integrates endoscopic cameras and sensitive force sensors, allowing surgeons to perform remote surgeries and monitor the entire process, even feeling the cutting or gripping force of the robotic arms on the patient.

Figure 1.4: Surgical Robot Da Vinci

1.2 Human - Robot Collaboration (HRC)

With advancements in science and technology, robots today are no longer the bulky, rigid machines associated with heavy and predefined tasks. By applying breakthroughs in sensors and artificial intelligence, robots are becoming increasingly dexterous and intelligent. Robotic systems can collect data from their surroundings, process and evaluate it, and then make autonomous decisions for their actions. These systems are being applied across various sectors and fields in collaborative roles. The integration of robots with human workers brings numerous benefits, such as increased productivity and reduced physical and mental workload for humans. Consequently, collaborative robots, or Cobots, have emerged as a globally focused research area.

1.2.1 What is Cobot?

Collaborative robots, or Cobots are robots equipped with capabilities to interact with humans in work. The abilities to interact and collaborate are two key aspects that developers focus on researching and developing. Human-Robot Interaction (HRI) refers to the robot's ability to respond to human behavior through communication channels such as vision and voice, allowing the robot to adjust its actions accordingly. Human-Robot Collaboration (HRC), on the other hand, refers to the robot's ability to support and enhance the mutual benefits achieved by both humans and robots, such as increasing productivity and reducing human errors while assuring safety in the procedure. Thus, it can be understood that interaction capability is the foundation for collaboration capability.

Figure 1.5: Universal Robots models

1.2.2 Interaction taxonomy

Due to the characteristics and cognitive abilities of the robots, HRI is defined differently by researchers. In the scope of this project, we will propose a taxonomy on the definition of Human Robot Interaction.

- **Cell**: Robots and humans operate and work in separate environments without any overlap. In this scenario, humans and robots perform different tasks and do not share a common goal. This is evident in large industrial factories where fully automated production lines are present.
- **Coexistance** : Robots and humans do not work simultaneously but subsequently, and their working environments do not intersect. Typically, the output of one entity (the robot) serves as the input for the other entity (the human).
- **Synchronized**: Robots and humans works and operates in the same workspace, but only one entity works at a given time, meaning that human and robot take turns in working.
- **Cooperation**: Robots and humans works and operates in the same time, sharing the workspace but will not work on the same problems, instead working in different problems to accomplish a bigger task.
- **Collaboration**: Robots and humans work simultaneously, sharing a common goal on a component within the same working environment.

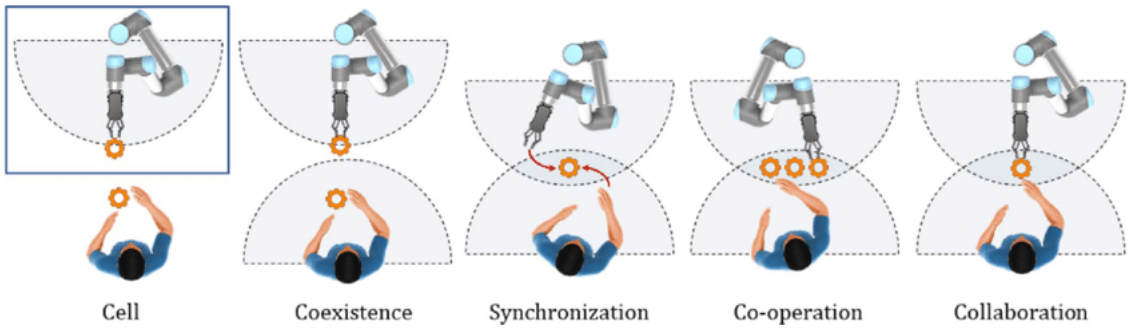

Figure 1.6: Human - Robot Collaboration taxonomy

1.2.3 Some recent researches

Roboticists in recent times has been researched and made some progress on the matter, proposing new methods for improving and enhancing the HRI and HRC behaviors. Researches on collaborations are predominating, where the tasks performed by humans and machines together are often complex processes that can be divided into sequential steps. Robots are tasked with assisting users in completing each step. In **[21]**,Robots are trained using Markov chains to assist humans in assembling boxes, by understanding the sequence of human actions and the meaning behind the executed action sequences. In some cases, robots can predict human actions, trajectories, and behaviors. In **[10]**, robot predicts the trajectory of the human body, and giving the according response base on its guess of the human action.

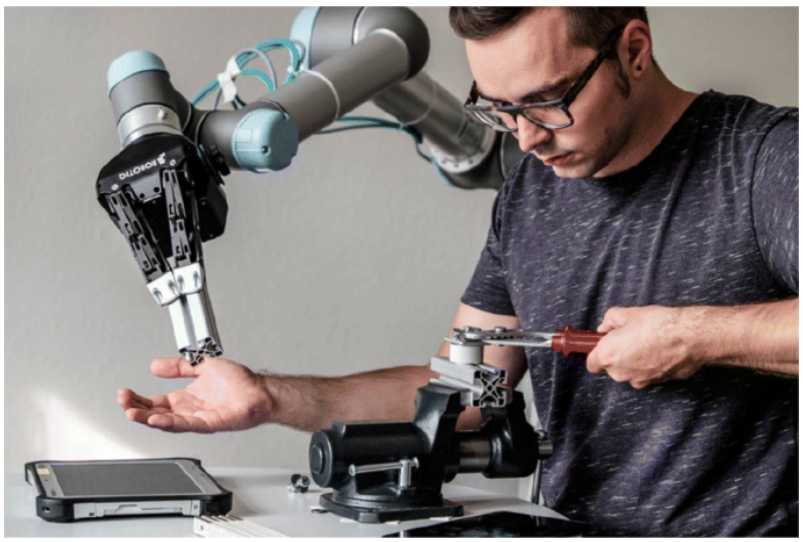

Figure 1.7: Collaborative robot in industry

In addition, numerous studies have focused on the safety aspects of human-machine interaction, concentrating on the behavior and responses of robots. In **[14]**,The authors have developed a model to predict the motion and trajectory of the human arm, which is then used to command the robot to plan an optimal path. These studies focus on addressing collision problems, emphasizing safety, and minimizing risks to humans and robot malfunctions. Or in **[29]**, the authors ultilizing the impedance controller to control the manipulator to "listen" to human behaviors when the collision between the two occures. Researching on dynamic path planning is also a hot topic, with considering human as a dynamic obstacle,**[20]** proposed a planner with the ability to avoid these obstacles and still reaches the goal destination.

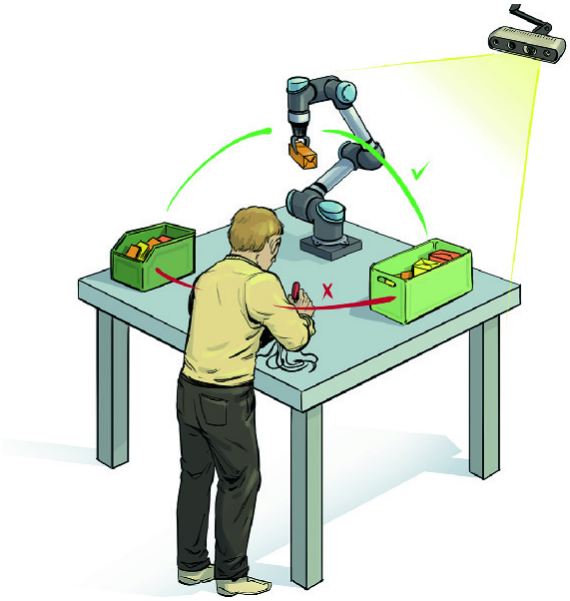

Figure 1.8: Dynamic obstacle avoidance in shared workspace

1.3 Project Proposal

1.3.1 Research goals

The breakthrough development of artificial intelligence systems has opened up new opportunities and areas in image processing. The use of deep learning models or convo-

lutional neural networks has accelerated image processing and demonstrated the potential of these models when applied to automated systems and robots.

The combination of advanced artificial intelligence models with robotic manipulators has created opportunities for the birth and development of intelligent and safe automation systems in the industrial production environment. Artificial intelligence provides robots with the ability to accurately perceive, from models that help robots recognize specific factors in the environment to controlling robots to make decisions in situations that are not pre-programmed. If this combination can be effectively exploited, an automated robot system can easily operate alongside humans in a real-world work environment.

The goal of this work is to develop a robotic arm system that is capable of avoiding collisions with humans in the shared work environment. To achieve this, we will focus on the following key aspects:

- Developing a vision system that can detect humans and predicts human's position in the workspace.
- Developing a path planner that consists of a global planner and local planner based on Rapidly-exloring Random Tree (RRT) and Artificial Potential Fields (APF) methods.
- Assuring the safety of human in human - robot collaboration task.

1.3.2 Project proposal

In the brainstorming process, our group has find the reseach of **[2]** ,applying the Artificial Potential Fields (APF) method into path planning procedure in order to create the object avoiding behaviors for the manipulator. The authors have proposed a way to approach APF - an method of creating attracting and repulsing forces fields to plan a suitable path, as well as a way to overcome the local minima problem - the common problem of the APF method.

Finding the topic interesting, we picked it as the base idea for the project. This work will focus on harnessing the "safety" aspect of path planning in case of dynamic obstacles; proposing a new hybrid planner that combines RRT and APF; experimenting and evaluating the result from each planners in a certain scenario.

The content of the project will be divided as:

- **Chapter 1**: Project overview
- **Chapter 2**: Robotic manipulator kinematics
- **Chapter 3**: Robot vision module
- **Chapter 4**: Robot path planning
- **Chapter 5**: Conclusion

1.4 Experiment scenario

To evaluate the planner,we conduct our experiments on the scenario as seen in 1.9, comprises of:

- UR5 robot with Robotiq Gripper mounted.
- Camera Intel RealSense D435.
- An open box as a known obstacle.
- Table, jigs .

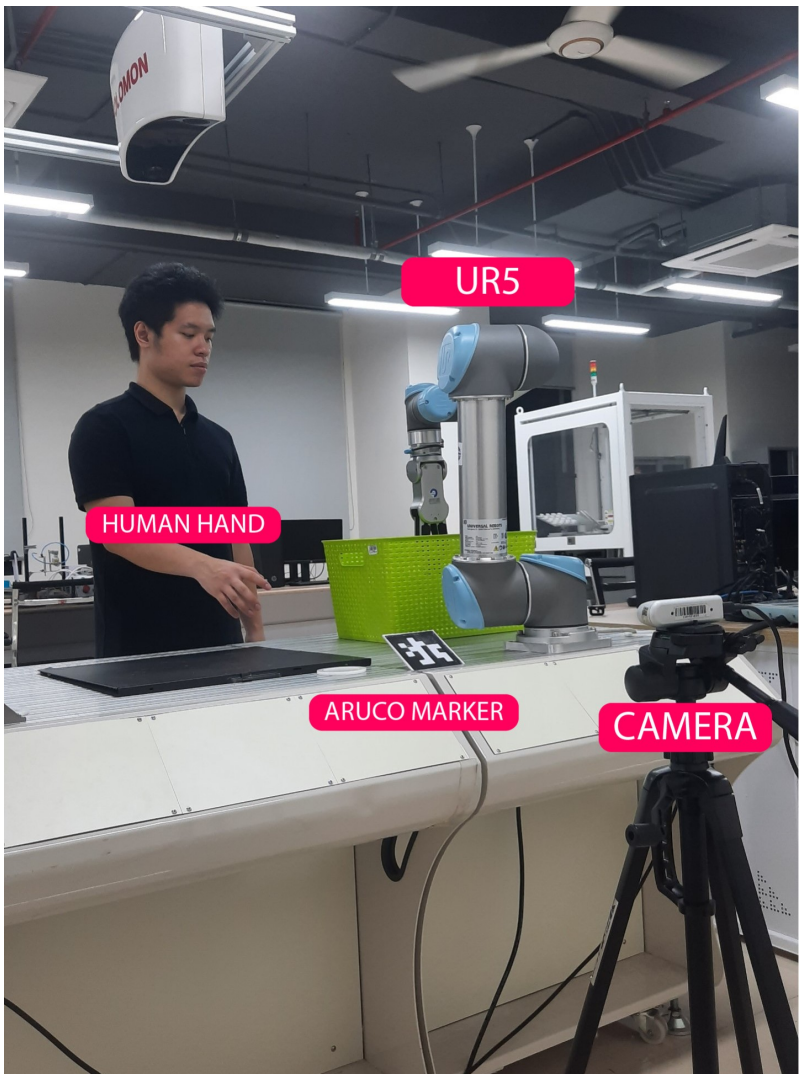

Figure 1.9: The experiment scenario

With:

- *Reference Path*: Path and trajectory generated by the global planner.
- *Dynamic Path*:Path and trajectory of the robots when encountering human hand in the shared workspace.

2.1 UR5 Manipulator

Universal Robot (UR) is one of the top robot manipulator manufacturers in the world, famously with the line of collaboration robots (Cobot). UR has set the new trend in the manipulator field of research with dexterous, safe and its simple deployability. UR's robots are distinguished not only by their safe collaboration capabilities with humans but also by their simple programming, which reduces complexity in the integration process into production environments. With their flexibility and adaptability, Universal Robots have become a superior solution for businesses looking to optimize productivity and flexibility in their production processes.

Figure 2.1: UR robot lines

Found in 2005 in Denmark, comparing to other manifacturers, UR is still considered new to the fields. However, However, the company's strength lies in its consistency from technology to design. UR focuses exclusively on multi-degree-of-freedom collaborative robotic arms but develops them at various levels to meet different needs and optimize costs for customers. Currently, UR has perfected two main robot lines: CB3 and E-Series.

The UR5 is a collaborative robotic arm from the CB3 line, providing a flexible solution for light and medium-duty tasks in production. With 6 joints, it can move similarly to a human arm.

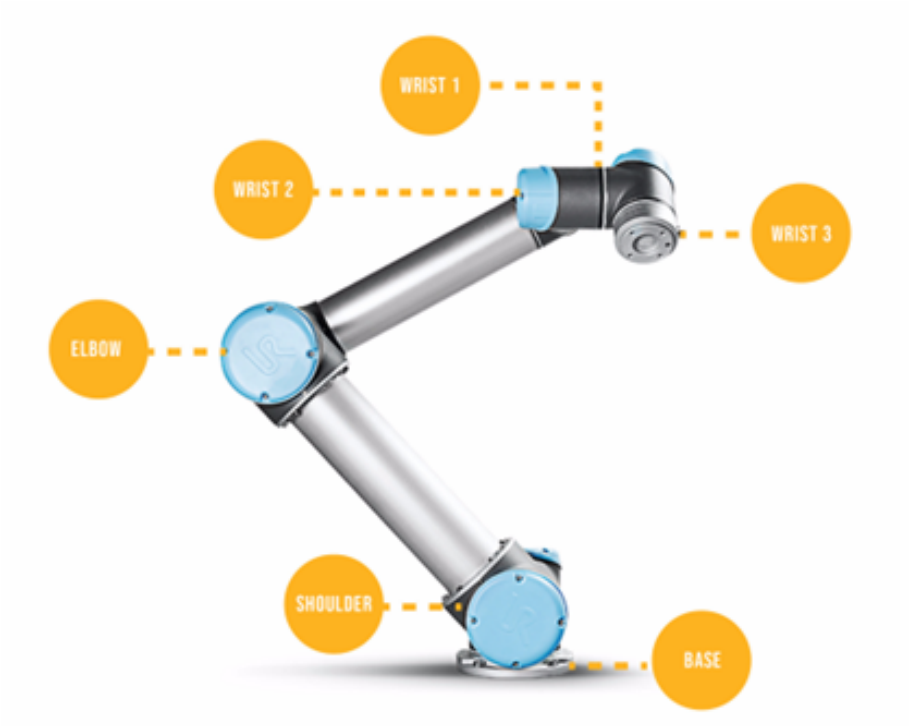

Figure 2.2: Robot UR5

2.1.1 Geometrical Properties

Table 2.1: Technical Specification UR5

Degree of Freedom	6
Reachability	850 mm
Maximum Load	5 kg
Maximum Power	570 W
Average Power	200 W
Operating Temperature	0 - 50°C
Material	Aluminum, Plastic, Steel
Base Parameter	149 mm
Digital Input	2
Digital Output	2
Analog Input	2
Analog Output	2

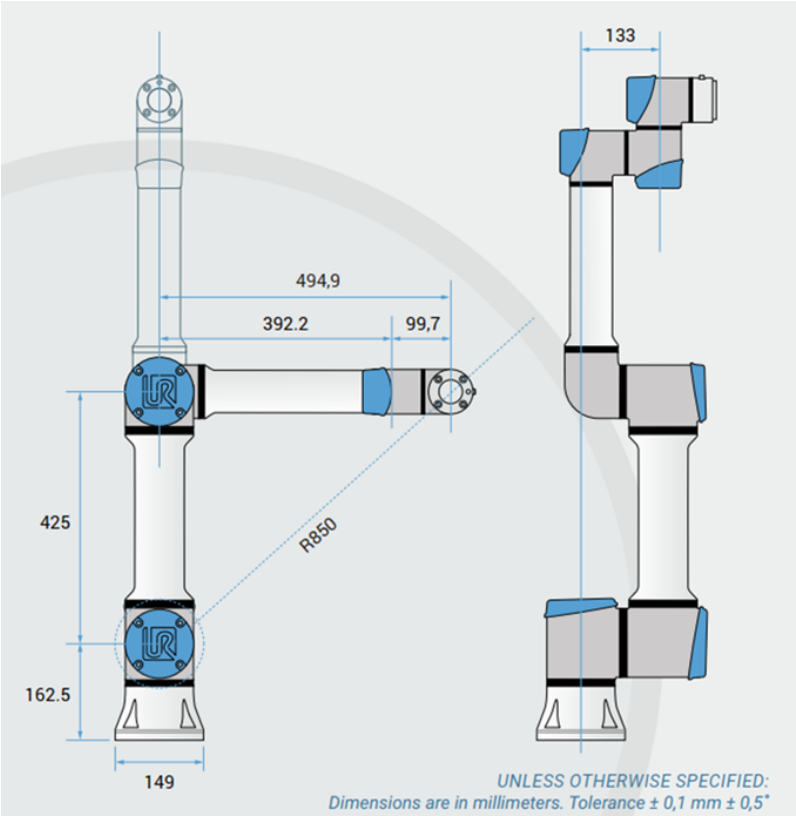

Figure 2.3: UR5 size

Table 2.2: Working Range

No.	Axis Movement	Min	Max
1	Base	-360°	360°
2	Shoulder	-360°	360°
3	Elbow	-360°	360°
4	Wrist 1	-360°	360°
5	Wrist 2	-360°	360°
6	Wrist 3	-360°	360°

Table 2.3: Mass and Center of Joints

Joints	Mass	Centers
1	3.7	[0, -0.02561, 0.00193]
2	8.393	[0.2125, 0, 0.11336]
3	2.33	[0.15, 0, 0.0265]
4	1.219	[0, -0.0018, 0.01634]
5	1.219	[0, 0.0018, 0.01634]
6	0.1879	[0, 0, -0.001159]

2.1.2 Workspace

The workspace of the robot is approximately 850mm from its joints, however the manifacturer only recommend operating under the range of 750mm to avoid unnecessary mistakes. We need to pay attention to considering the circular space directly above and below

the base of the robot when choosing an installation location for the robot. Putting the working tool close to the cylindrical area should be avoided because it causes the base joints to move quickly even when the tool moves slowly, which makes the robot work inefficiently and causes difficulty .

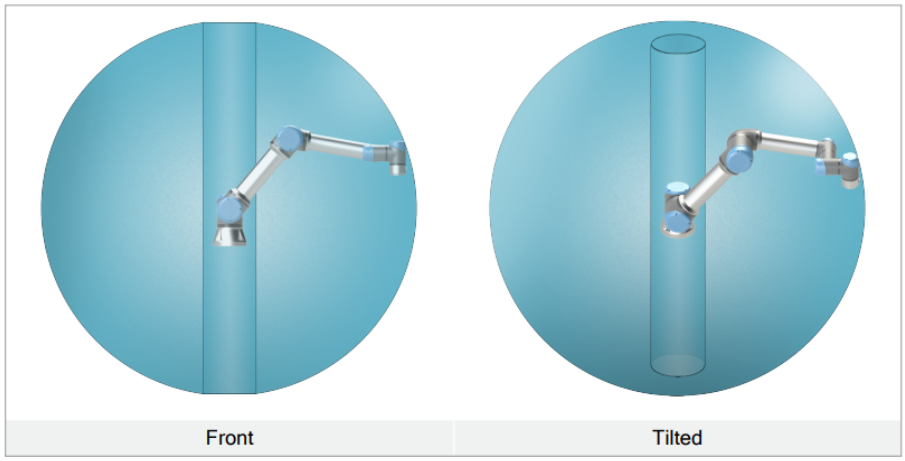

Figure 2.4: UR5 workspace

2.2 Robotiq Hand-E Gripper

Robotiq is a leading manifacturer in automation, specilized in developing solutions for robots. In their catalog, Gripper Hand-E stands out as a dexterous, flexible and can be easily intergrated with many robot models. Hand-E is not only built for optimizing the automation process, but also provides high flexibity for industrial applications.

Figure 2.5: Robotiq Hand-E Gripper

The ingenious design makes Hand-E a perfect solution in multiple automation and robot lines. Especially to UR manipulator lines, with the support of mounting piece and URCaps (driver) built for these external grippers. The table below shows the technical properties of Hand-E Gripper.

Stroke	50mm
Form-fit Grip Payload	5kg
Grip Force	20-185N
Closing speed	20-150mm/s

Table 2.4: Robotiq Hand-E Gripper Specification

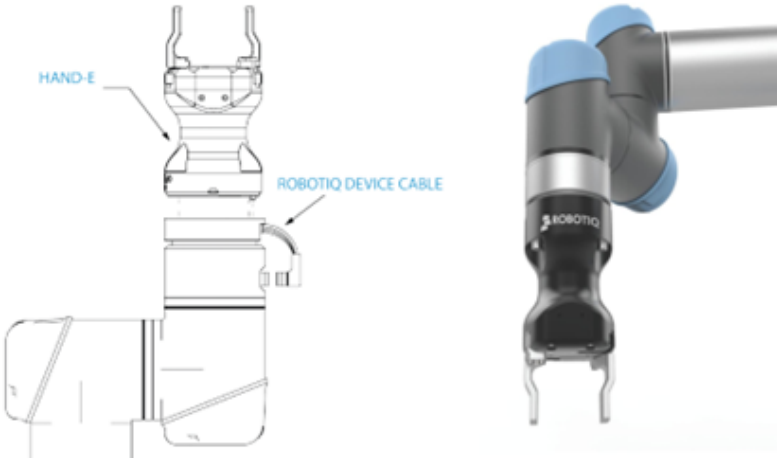

Figure 2.6: Mounted Hand-E Gripper on UR5

2.3 Denavit - Hartenberg Method

Denavit-Hartenberg parameters (D-H) is a method widely used in robotics to calculate kinamtics problems, by using four parameters associated with with a particular convention for attaching reference frames to the links of a spatial kinematic chain, or robot manipulator.

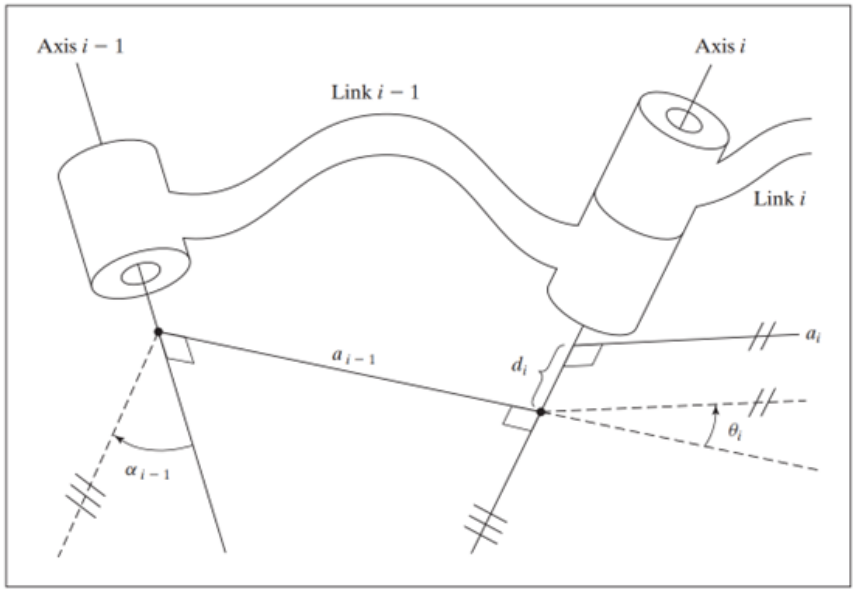

Figure 2.7: Denavit-Hartennberg Parameters

According to the convention that Jacques Denavit and Richard Hartenberg introduced in 1955, the standardize frames i can be described with :

- Axis z(i) is the axis of joint (i+1), is in the direction of the joint axis, connecting link (i) to link (i+1).
- O(i) is the intersection point of z(i) with the orthogonal lines of z(i) and z(i-1).
- Axis x(i) is the orthogonal lines of z(i-1) and z(i), in the direction of z(i-1) to z(i)
- Axis y(i) is picked accordingly to the user.

The four parameters are:

(a):Length of the common normal. Assuming a revolute joint, this is the radius about previous z.

(α) : Angle about common normal, from old z axis to new z axis

(d): Offset along previous z to the common normal.

(θ): Angle about common normal, from old z axis to new z axis

By assigning these parameters to each link, a set of transformation matrices can be derived to describe the position and orientation of each link relative to the base frame. These matrices are used to perform forward and inverse kinematics calculations, enabling the robot to accurately position and orient its end effector.

2.4 Kinematics

2.4.1 UR5 coordinates system

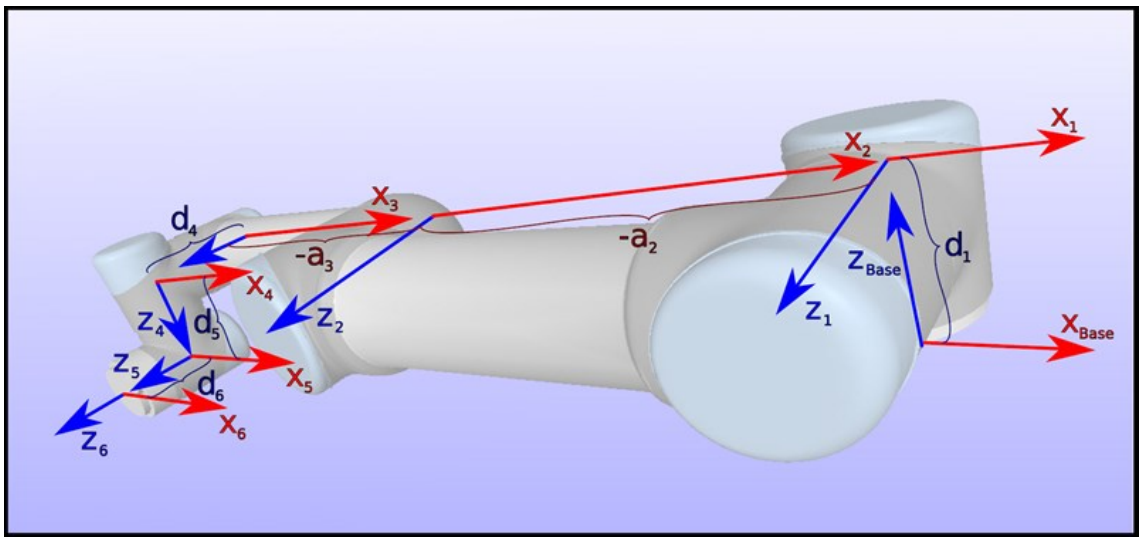

Figure 2.8: UR5 D-H Parameters

- Red: X axis
- Blue: Z axis

2.4.2 UR5 D-H Table

Link	$\theta_i(rad)$	$d_i(m)$	$\alpha_i(rad)$	$a(m)$
1	θ_1	0.1625	π/2	0
2	θ_2	0	0	0.6127
3	θ_3	0	0	0.57155
4	θ_4	0.1333	π/2	0
5	θ_5	0.0997	-π/2	0
6	θ_6	0.0996	0	0

The transformation matrices from link (i-1) to link (i) from the parameters of D-H is described as:

$$^{i-1}\boldsymbol{T_i} = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

The UR5 transformation matrix is:

$$^{0}\boldsymbol{T_6} = {}^0T_1\,{}^1T_2\,{}^2T_3\,{}^3T_4\,{}^4T_5\,{}^5T_6 \tag{2.2}$$

With the transformation matrix from 0 to 1 coordinates:

$$^0T_1 = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & 0.1625 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

The transformation matrix from 1 to 2 coordinates:

$$^1T_2 = \begin{bmatrix} c_2 & -s_2 & 0 & a_2c_2 \\ s_2 & c_2 & 0 & a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_2 & -s_2 & 0 & -0.425c_2 \\ s_2 & c_2 & 0 & -0.425s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

The transformation matrix from 2 to 3 coordinates:

$$^2T_3 = \begin{bmatrix} c_3 & -s_3 & 0 & a_3c_3 \\ s_3 & c_3 & 0 & a_3s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_3 & -s_3 & 0 & -0.3922c_3 \\ s_3 & c_3 & 0 & -0.3922s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

The transformation matrix from 3 to 4 coordinates:

$$^3T_4 = \begin{bmatrix} c_4 & 0 & s_4 & 0 \\ s_4 & 0 & -c_4 & 0 \\ 0 & 1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_4 & 0 & s_4 & 0 \\ s_4 & 0 & -c_4 & 0 \\ 0 & 1 & 0 & 0.1333 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

The transformation matrix from 4 to 5 coordinates:

$$^4T_5 = \begin{bmatrix} c_5 & 0 & -s_5 & 0 \\ s_5 & 0 & c_5 & 0 \\ 0 & -1 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_5 & 0 & -s_5 & 0 \\ s_5 & 0 & c_5 & 0 \\ 0 & -1 & 0 & 0.0997 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

The transformation matrix from 5 to 6 coordinates:

$$ {}^{5}T_{6} = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & 0.0996 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.8} $$

Using MATLAB, we can get the transformation matrix from 0 to 6 coordinates:

$$ {}^{0}T_{6} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & P_x \\ r_{21} & r_{22} & r_{23} & P_y \\ r_{31} & r_{32} & r_{33} & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.9} $$

The position of the end-effector (EE) can be calculated as:

$$ P_x = d_4 s_1 + a_2 c_1 c_2 + d_6(c_5 s_1 - c_{234} c_1 s_5) + d_5 s_{234} c_1 + a_3 c_1 c_2 c_3 - a_3 c_1 s_2 s_3 \tag{2.10} $$

$$ P_y = -d_4 c_1 + a_2 c_2 s_1 - d_6(c_1 c_5 + c_{234} c_1 s_5) + d_5 s_{234} s_1 + a_3 c_2 c_3 s_1 - a_3 s_1 s_2 s_3 \tag{2.11} $$

$$ P_z = d_1 + a_3 s_{23} + d_5(s_{23} s_4 - c_{23} c_4) + a_6 s_5 (s_{23} s_4 + c_{23} c_4) \tag{2.12} $$

The orientation of the EE can be described using the orientation matrix:

$$ \begin{bmatrix} c_6 s_1 s_5 + c_1 c_5 c_{234} - c_1 s_6 s_{234} & -s_6 s_1 s_5 + c_1 c_5 c_{234} - c_1 c_6 s_{234} & c_5 s_1 - c_1 s_5 c_{234} \\ -c_6 c_1 s_5 - c_5 s_1 c_{234} - s_1 s_6 s_{234} & s_6 c_1 s_5 - c_5 s_1 c_{234} - c_6 s_1 s_{234} & -c_1 c_5 - s_1 s_5 c_{234} \\ s_6 c_{234} + c_5 c_6 s_{234} & c_6 c_{234} - c_5 s_6 s_{234} & -s_5 s_{234} \end{bmatrix} \tag{2.13} $$

2.5 Inverse Kinematics

After solving the forward kinematics problem, we proceed to solve the problem of determining the joint variables given the position and orientation of the end-effector. This is the inverse kinematics problem, which is considered more challenging than the forward kinematics problem. The complexity of the problem depends on the structural configuration of the robot and the degrees of freedom to derive a reasonable solution.

Solving the inverse kinematics problem requires finding solutions to a system of nonlinear algebraic equations. Therefore, the problem may have multiple solutions, a single solution, or no solution at all. Typically, there are two methods to solve this: the geometric method and the analytical method. In this thesis, we will choose the geometric method for convenient calculations.

Assume that we know the orientation and position of the end-effector as the rotation matrix $\boldsymbol{{}^{0}T_{6}}$ and the position vector $\boldsymbol{P_6}$.

We can find the translation vector from the base coordinates to link (5) coordinates:

$$ {}^{0}P_5 = {}^{0}P_6 + {}^{0}R_6 * [0, 0, -d_6]^T \tag{2.14} $$

With the translation vector (5) ${}^{0}P_5$, we can find θ_1.Because the rotating motion around the axis z_1, z_2, z_3, z_4 does not affect the fact that the coordinates origin 5 is on the plane that parallel to the x_1 axis:

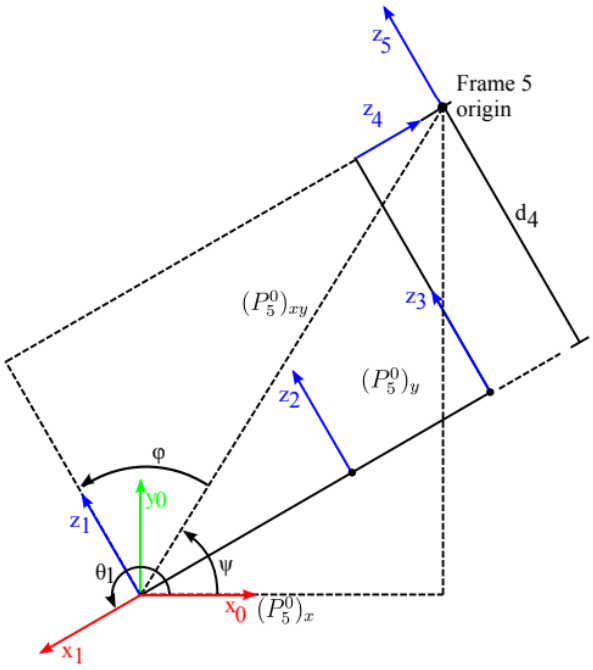

Figure 2.9: Finding θ_1

We can calculate $\boldsymbol{\theta_1 = \psi + \phi + \pi/2}$, with:

$$\psi = atan2((^0P_5)_y, (^0P_5)_x) \tag{2.15}$$

$$\phi = \pm \arccos(\frac{d_4}{(^0P_5)_{xy}}) \tag{2.16}$$

$$= \pm \arccos(\frac{d_4}{(\sqrt{(^0P_5)_x^2 + (^0P_5)_y^2}}) \tag{2.17}$$

The set of equations yield 2 solution of θ_1 corresponding to the case that shoulder joint is in the "left" or "right" to the link. With that the equation 2.16 is always true with the condition $d_4 > (^0P_5)_{xy}$.

We continue to find θ_5. Considering the the coordinate origin 6 to the coordinate origin 1.

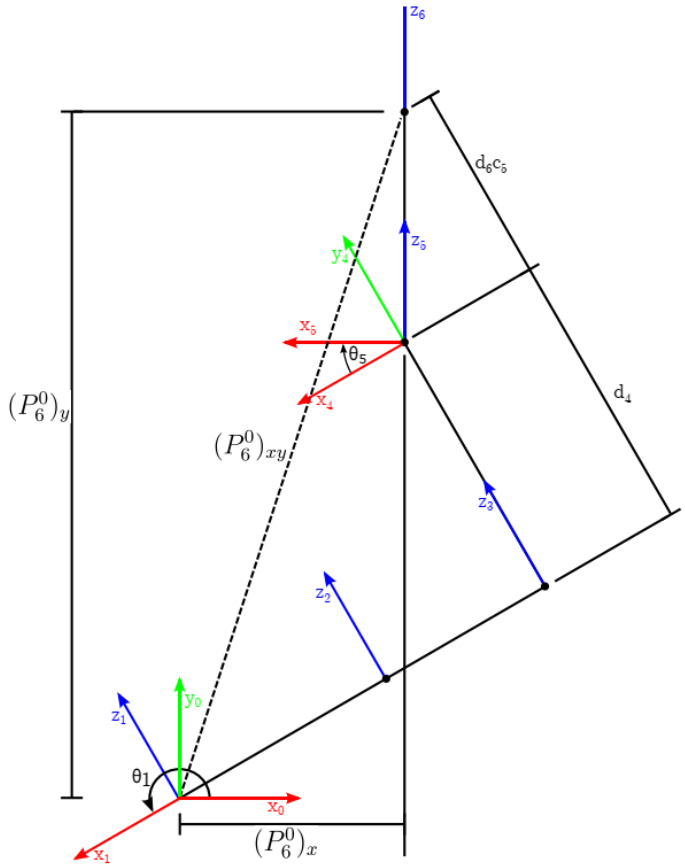

Figure 2.10: Finding $theta_5$

From figure 2.5, we can calculate $({}^1P_6)_z = d_6\cos(\theta_5)+d_4$ with $({}^1P_6)_z = ({}^0P_6)_x\sin(\theta_1)-({}^0P_6)_y\cos(\theta_1)$. θ_5 can be solved as :

$$\theta_5 = \pm\arccos(\frac{({}^1P_6)_z - d_4}{d_6}) \tag{2.18}$$

Two solution of θ_5 corresponding to two "upper" and "under" positions.

To find θ_6, we have to find the transformation 6 coordinates to 1 coordinates.

$${}^6T_1 = (({}^0T_1)^{-1}({}^0T_6))^{-1} \tag{2.19}$$

The relation between T_{31}, T_{32} is the value of each elements in column 3 of the matrix:

$$T_{32} = -\sin(\theta_6)\sin(\theta_5) \tag{2.20}$$
$$T_{31} = \cos(\theta_6)\cos(\theta_5) \tag{2.21}$$

From that, we can calculate θ_6:

$$\theta_6 = atan2(\frac{-T_{32}}{\sin(\theta_5)}, \frac{T_{31}}{\sin(\theta_5)}) \tag{2.22}$$

We can find the other joints' values due to they form a RRR rigid tree on the same plane. From that, we find the translation vector from 3 coordinates to 1 coordinates.

$${}^1T_4 = {}^1T_6{}^6T_4 = {}^1T_6({}^4T_5{}^5T_6)^{-1} \tag{2.23}$$
$${}^1P_3 = {}^1R_4[0, -d_4, 0]^T \tag{2.24}$$

We have the following relations:

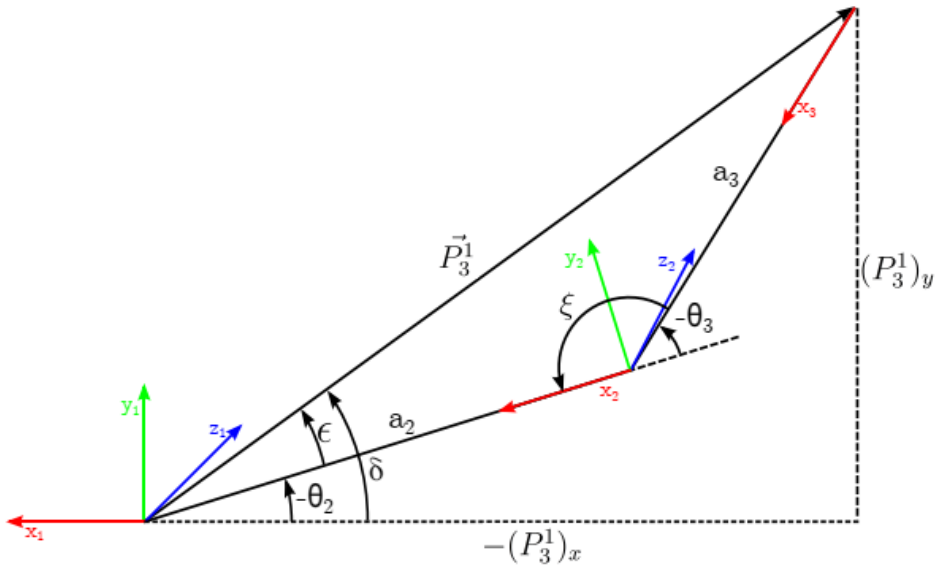

Figure 2.11: Finding θ_2 and θ_3

Find θ_3:

$$\cos(\theta_3) = \cos(\xi) = \frac{||^1\vec{P}_3||^2 - {a_2}^2 - {a_3}^2}{2a_2a_3} \tag{2.25}$$

From the figure we got $\theta_2 = -(\delta - \epsilon)$, with:

$$\delta = atan2({}^1P_{3y}, -{}^1P_{3x}) + \arcsin(\frac{a_3 \sin(\theta_3)}{||^1\vec{P}_3||}) \tag{2.26}$$

Finally, we can find θ_4 by finding the matrix ${}^3T_4$:

$${}^3T_4 = {}^3T_1 {}^1T_4 = {}^1T_2 {}^2T_3 {}^{-1}{}^1T_4 \tag{2.27}$$

Using the 2 elements from the first column, we got:

$$\theta_4 = atan2(T_{21}, T_{11}) \tag{2.28}$$

2.6 Velocity and Acceleration

2.6.1 Angular velocity and acceleration

The angular velocity ω characterizes the change in the orientation of an object along three directions in the coordinate system. We can calculate it using the time derivative of the vectors belonging to the object:

$$\tilde{\omega^0} = \dot{R}R^T = \begin{bmatrix} 0 & -\omega_z^{(0)} & \omega_y^{(0)} \\ \omega_z^{(0)} & 0 & -\omega_x^{(0)} \\ -\omega_y^{(0)} & \omega_x^{(0)} & 0 \end{bmatrix} \tag{2.29}$$

with $\omega^{(0)} = [\omega_x^{(0)}, \omega_y^{(0)}, \omega_z^{(0)}]^T$ is the angular velocity vector in the base coordinates.

We can find the angular acceleration $\alpha^{(0)}$ in the base coordinates:

$$\alpha^{(0)} = \dot{\omega}^{(0)} \quad = \begin{bmatrix} \dot{\omega}_x^{(0)} \\ \dot{\omega}_y^{(0)} \\ \dot{\omega}_z^{(0)} \end{bmatrix} \tag{2.30}$$

2.6.2 Translational velocity and acceleration

We can find the translation vector between a base point O and a random P:

$$r_P^{(0)} = r_O^{(0)} + Ru^{(i)} \tag{2.31}$$

with $u^{(i)}$ is the vector going through P in the base coordinates.

Taking the derivative of the previous equation, we can find the velocity of P:

$$\dot{r_P^{(0)}} = \dot{r_O^{(0)}} + \dot{R}u^{(i)} \tag{2.32}$$

Or:

$$v_P^{(0)} = v_O^{(0)} + \tilde{\omega^0} Ru^{(i)} \tag{2.33}$$

2.6.3 Jacobian Matrix

The Jacobian matrix represents the relationship between velocity and angles in task space and joint space. The Jacobian of a vector-valued function in several variables is the matrix of all its first-order partial derivatives. This quantity is extremely important in analyzing and controlling the motion of robots.

The Jacobian matrix can be expressed by the equation:

$$[J] = \begin{bmatrix} \dfrac{\partial f_i}{\partial \theta_j} \end{bmatrix} \tag{2.34}$$

with f_i is the equations describing the pose of the robot, and θ_i is the joints' variables. Jacobi matrix is created by two components, translational and rotational, J_T and J_R.

2.6.4 Calculate Jacobian matrix from Homogenous matrix

After solving the forward kinematics problem, we obtain the position and orientation of the end-effector relative to the fixed frame 0 through the homogeneous transformation matrix:

$$T_n^0 = \begin{bmatrix} R_n^0 & r^(0)_O n \\ 0 & 1 \end{bmatrix} \tag{2.35}$$

From there, we can calculate the linear velocity and angular velocity of the end-effector, where v_E^0 - the translational velocity vector of the end-effector in the base frame is calculated as follows:

$$\mathbf{v}_E^{(0)} = \frac{d}{dt}\mathbf{r}_E^{(0)} = \frac{\partial \mathbf{r}_E^{(0)}}{\partial \mathbf{q}}\dot{\mathbf{q}} = \mathbf{J}_T(\mathbf{q})\dot{\mathbf{q}} \tag{2.36}$$

$\mathbf{J}_T$ is the translational Jacobian Matrix, can be calculated as:

$$\mathbf{J_T}(\mathbf{q}) = \frac{\partial \mathbf{r}_E^{(0)}}{\partial \mathbf{q}} = \begin{bmatrix} \frac{\partial x_E^{(0)}}{\partial q_1} & \frac{\partial x_E^{(0)}}{\partial q_2} & \cdots & \frac{\partial x_E^{(0)}}{\partial q_n} \\ \frac{\partial y_E^{(0)}}{\partial q_1} & \frac{\partial y_E^{(0)}}{\partial q_2} & \cdots & \frac{\partial y_E^{(0)}}{\partial q_n} \\ \frac{\partial z_E^{(0)}}{\partial q_1} & \frac{\partial z_E^{(0)}}{\partial q_2} & \cdots & \frac{\partial z_E^{(0)}}{\partial q_n} \end{bmatrix} \tag{2.37}$$

In our project, UR5 has 6 joints, translational Jacobian matrix $\mathbf{J}_T$ will be a 6×3 matrix.

From the transformation matrix, we can calculate the angular velocity of the final link as in 2.29. Due to the linear depedance of angular velocity to the velocity of the joints $\dot{q}$, we can write it as follows:

$$\omega_n^{(0)} = J_R(q)\dot{q} \tag{2.38}$$

Rotational Jacobian matrix $J_R(q)$ can be calculated as:

$$\mathbf{J_R}(\mathbf{q}) = \frac{\partial \mathbf{r}_E^{(0)}}{\partial \mathbf{q}} = \begin{bmatrix} \frac{\partial \omega_{n,x}^{(0)}}{\partial \dot{q_1}} & \frac{\partial \omega_{n,x}^{(0)}}{\partial \dot{q_2}} & \cdots & \frac{\partial \omega_{n,x}^{(0)}}{\partial \dot{q_n}} \\ \frac{\partial \omega_{n,y}^{(0)}}{\partial \dot{q_1}} & \frac{\partial \omega_{n,y}^{(0)}}{\partial \dot{q_2}} & \cdots & \frac{\partial \omega_{n,y}^{(0)}}{\partial \dot{q_n}} \\ \frac{\partial \omega_{n,z}^{(0)}}{\partial \dot{q_1}} & \frac{\partial \omega_{n,z}^{(0)}}{\partial \dot{q_2}} & \cdots & \frac{\partial \omega_{n,z}^{(0)}}{\partial \dot{q_n}} \end{bmatrix} \tag{2.39}$$

In our project, UR5 has 6 joints, rotational Jacobian matrix $\mathbf{J}_R$ will be a 6×3 matrix.

The robot's Jacobian matrix can be written as follows:

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} J_T(q) \\ J_R(q) \end{bmatrix} \tag{2.40}$$

3.1 System components

3.1.1 Camera Intel Realsense D435

Intel RealSense Depth Camera D435 is an advance 3D camera from Intel, designed to capturing and providing precise RGB and depth images. With compact design and cross-platform connectivity, D435 is a ideal choice for vision solution. In this thesis, the camera is used for human detection, from that we can calculate the position of the human (wrist to be exact) to plan the new path.

Figure 3.1: Intel RealSense Depth Camera D435

Left and right imager sensor	
Imager Sensor	OV9282
Active pixels	1280×800
Sensor aspect ratio	8:5
Output format	10 - bit RAW
F number	f/2.0
Focal Length	1.93 mm
Focus	Fixed
Horizontal FOV	91.2°
Vertical FOV	65.5°
Diagonal FOV	100.6°
Distortion	$\leq$ 1.5%
Camera RGB	
Imager Sensor	OV2740
ISP	Discrete
Active pixels	1920×1080
Sensor aspect ratio	16:9
Format	10 - bit RAW RGB
F number	f/2.0
Focal Length	1.93 mm
Focus	Fixed
Horizontal FOV	69.4°
Vertical FOV	42.5°
Diagonal FOV	77°
Distortion	$\leq$ 1.5%

Table 3.1: Camera Specifications

3.1.2 OpenCV library - ROSBrigde

The OpenCV library is the open-source and one of the most well-known in image processing, providing functions for sending and receiving images as well as pre-processing procedure. In this project, OpenCV is utilized as the image processing library, facilitating the transmission and output of image data within the nodes of ROS.

Figure 3.2: OpenCV library

Because the data taken from ROS nodes are ROS messages $(sensor\_msgs/Image)$, to utilize OpenCV functions, we need to convert these images to Mat-like Image by using cv_bridge.

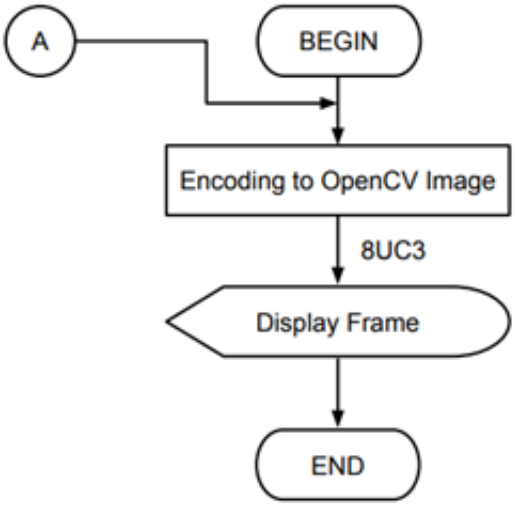

Figure 3.3: ROS - OpenCV brigde

The converstion only happened when it's called (the calling signal is A).

3.1.3 Aruco Marker

The Aruco Marker is an intelligent marker system developed to simplify the process of recognition and localization in the fields of computer vision and augmented reality. Resembled a QR code, the Aruco Marker is a square made up of black and white cells with a black border, create a matrix containing binary information about the size, position, and identity of the Aruco Marker.

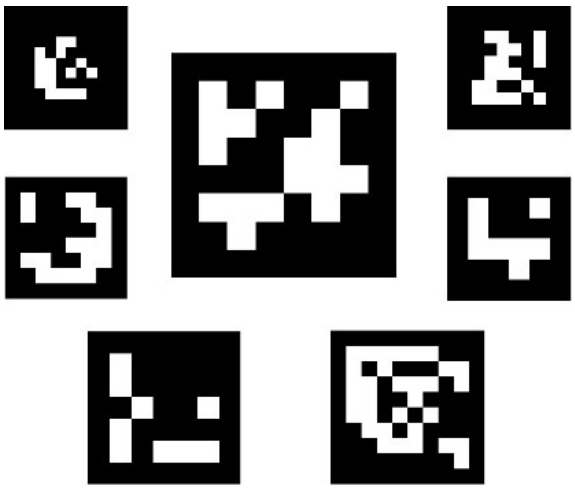

Figure 3.4: ArUco Markers

Markers are typically standardized as squares with varying sizes. This is because the black and white cells of the Marker create a characteristic binary matrix, allowing it to be recognized by commands. The ArUco library is responsible for initialization and definition, containing multiple libraries for different sizes. For example, an Aruco Marker of size 4x4 will contain 16 bits of data. In our project, we will use a 6x6 Aruco Marker.

3.1.4 A few classical methods

Binary Threshold

The image regions will be converted into black and white based on their brightness intensity being greater or less than a fixed threshold.

A drawback of this method is that it produces results that vary according to the intensity threshold chosen and often lacks accuracy for images without clearly defined color segments.

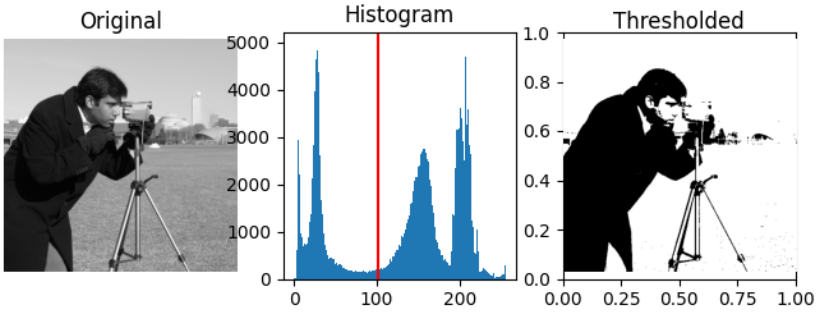

Figure 3.5: Example of binary threshold

K-mean clustering

The k-means clustering algorithm segments the intensity of pixels in an image into k clusters. Subsequently, each pixel's value is replaced by its nearest centroid to segment the image.

Disadvantages include: border regions between objects may be invaded, high computational cost due to the need to calculate distances from centroids to all pixels during training. Another limitation is the uncertainty in determining the appropriate number of clusters.

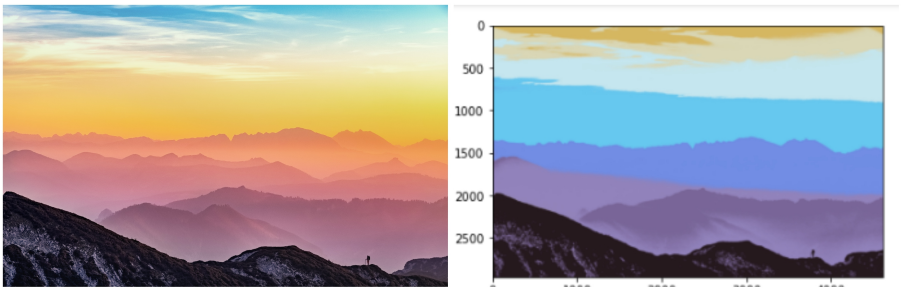

Figure 3.6: Example K-mean clustering

To tackle the task of segmenting humans in images, our team will focus on exploring recent advancements in artificial intelligence applied to image processing.

3.1.5 Mordern methods

In recent years, several modern architectures in deep learning have been developed to address the problem of Image Segmentation. Some prominent and modern architectures in this field include: U-Net, DeepLab, Mask R-CNN, FCN (Fully Convolutional Network), PSPNet (Pyramid Scene Parsing Network), among others.

These architectures are often combined with techniques such as transfer learning, augmentations, and specialized loss functions to improve model performance across various types of data. Each architecture has distinct features suited to different requirements and challenges. For instance:

- U-Net: Widely used in medical applications, such as cell segmentation in medical images like MRI or CT scans.
- DeepLab: Typically applied to object segmentation tasks in images captured from cameras and localization applications.
- Mask R-CNN: Often used for object detection and segmentation in images, suitable for applications like object recognition in images.

Mask R-CNN

Mask R-CNN is essentially an extension of Faster R-CNN. Faster R-CNN is widely used in object detection tasks. When an image is inputted into Faster R-CNN, it outputs labels and bounding boxes for each specific object in the image.

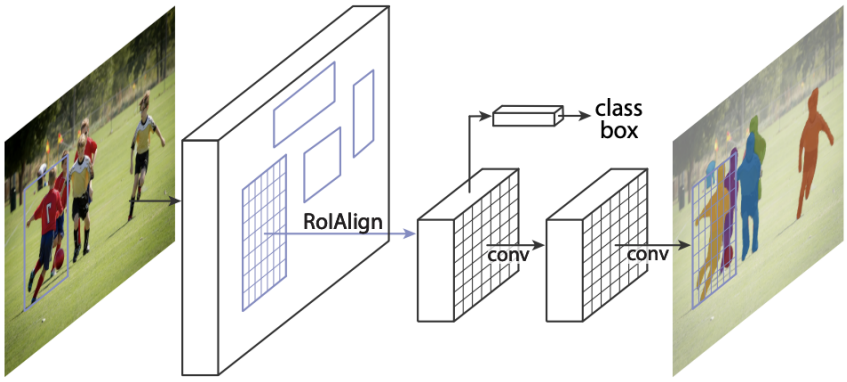

Figure 3.7: Architecture of Mask R-CNN

How the architecture works:

- First, a base network (or backbone) is a CNN that extracts image features. We truncate the output of the CNN and only take the feature map at the last layer.

- The feature map is then fed into a Region Proposal Network (RPN) to identify Regions of Interest (RoIs), which are areas likely to contain objects.
- RoIs are cropped from the feature map and processed through an RoI pooling layer to reshape and stack them into a uniform size.
- The RoI block is then passed through fully connected layers and split into two branches: one branch predicts the bounding box (BBox), and the other predicts the label.

In this research, thanks to the easy recognition capability of the Aruco Marker, we applied an Aruco Marker as an intermediary reference frame between the camera and the robot. This reduced the complexities associated with calculating coordinate system transformations, thereby making the problem simpler and more efficient.

3.2 Artificial Intelligence in image processing

Image processing or computer vision has long been an integral and indispensable field in automated robotic systems. Focusing on the processing and analysis of digital images, image processing includes various stages such as image information collection, data processing, noise processing, and the extraction and classification of derived parameters.

The most common problem in image processing is Image Classification. Here, the input is an image, and the output is its corresponding label. However, in some cases, multiple objects may appear in the input image. The task of image classification involves extracting the properties of objects in the image, bounding the objects in the image with a bounding box, and further creating masks that highlight those regions, known as Segmentation.

In our project, to create a safe interaction environment between humans and machines, recognizing and extracting information about people from the frame is our primary concern. First, images from the camera input will be processed and labeled for important objects in the frame. Once a person is recognized, we identify the ROI - the region of interest containing important information in the image. By processing the 3D point cloud of this region, we can obtain the central position of the human body parts.

3.2.1 YOLO characteristics

YOLO, or You Only Look Once, is an advance model in the field of detecting and classification of images. The YOLO model is designed to detect multiple objects in the same image frame fast and efficiently. This is a CNN (convolutional Neural Network) model. Some of the model characteristics are:

- (One-Shot):YOLO divides the image into a grid and performs the detection process once on the entire image, rather than splitting the image into many smaller parts like many traditional models.
- **Real-time efficiency**:YOLO is designed for real-time performance, making it suitable for applications such as object tracking, video surveillance, and use in autonomous vehicles.

Launched in 2015 and now reaching version 10.0, YOLO has quickly proven to be superior to other image processing models such as R-CNN or DPM. The version chosen by our team is YOLOv8, featuring improvements in the backbone and neck layers, including the use of new convolutional layers that combine convolution operations with shortcuts like the ResNet model. Moreover, YOLOv8 is an anchor-free model. An anchor box is a predefined rectangular shape with fixed size and aspect ratio. During training, the deep learning model predicts the coordinates and probability of the anchor box to detect objects in the image. When anchor boxes overlap, the probability of accurately detecting the target object increases. Instead, YOLOv8 is trained to directly recognize the center of the object, skipping intermediate steps such as offsetting from anchor boxes as in previous models.

During the training process, the deep learning model will predict the coordinates and probabilities of the anchor boxes to detect objects in the image. Using anchor boxes allows the model to predict objects at different positions and sizes.

Each object in the training image is assigned to an anchor box. In the case where two or more anchor boxes encompass the same object, the anchor box with the highest IoU (Intersection over Union) with the ground truth bounding box is selected.

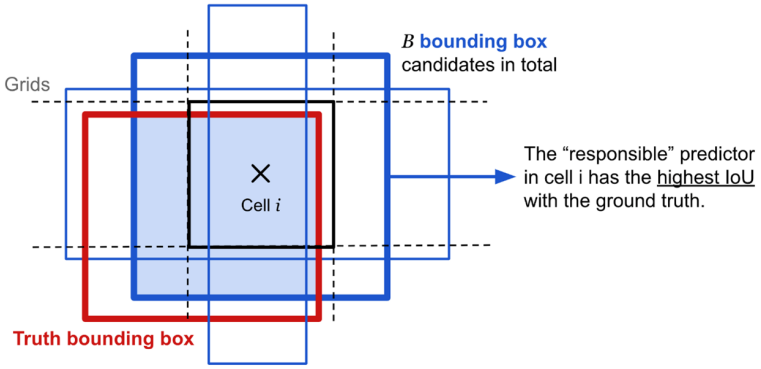

Figure 3.8: Finding the anchor box of an object

From Cell i, three anchor boxes with green borders are identified as shown in the image. All three anchor boxes intersect with the bounding box of the object. However, only the anchor box with the thickest green border is chosen as the anchor box for the object because it has the highest IoU compared to the ground truth bounding box. Each object in the training image is assigned to a cell on the feature map that contains the midpoint of the object.

Thus, when identifying an object, we need to determine two components associated with it: (cell, anchor box). It is not enough to consider just the cell or just the anchor box alone.

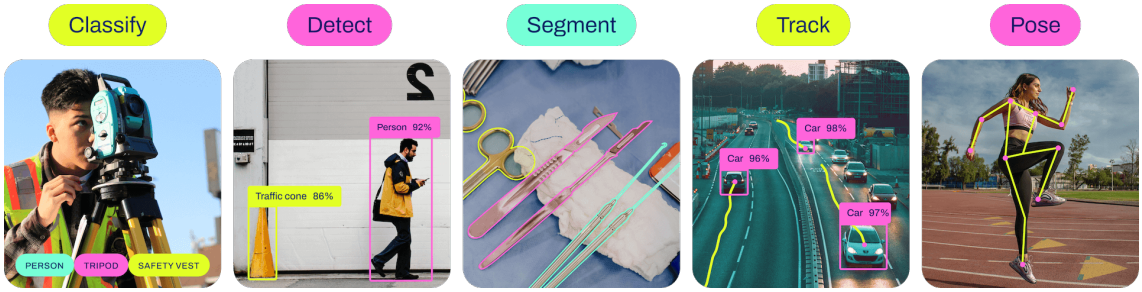

Figure 3.9: YOLOv8 Tasks

YOLOv8 is very flexible, and it can be utilized for tasks such as in 3.9.

Data	**Classification**	**Detection**	**Segmentation**	**Tracking**	**Pose Estimation**
Input	Image with objects needs to classified	Image with multiple objects	Image with one or multiple objects	Image with one or multiple objects	Image contains humans
Output	Classes' labels	Bounding boxes,labels	Determines which pixels belongs to which objects	Detects and maintains a unique ID for each detected object	Detects humans' body-parts pose and keypoints.

Table 3.2: YOLOv8 capable tasks

3.2.2 Common terms

Before we go into explaining the architecture and mechanism of the model, we need to clarify some common terms in machine learning.

- **Convolutional Neural Network**: a prominent method in machine learning, offers significant advantages in reducing computational load and improving accuracy compared to the Fully Connected architecture.
- **Maxpooling**: a technique for reducing the input size of data, selects the highest value regions in stacked layers, aiming to reduce the computational burden for the model.
- **Feature map**: LAs an output block from the CNN, it divides the output into a grid of square cells and applies object detection on each cell.
- **Epoch**: timestep that every sample in the training dataset has been used once to update the model's parameters.
- **Batch**:a subset of the training dataset used to train the model in one iteration. Instead of using the entire dataset to update the model's parameters, the dataset is divided into smaller batches.

3.2.3 YOLOv8 architecture and mechanism

YOLOv8 architecture are made of 3 parts:

- **Backbone**: The Deep Learning network is constructed to extract the features of images. This process is referred to as **feature extraction**. The backbone of YOLOv8 consists of a CNN (Convolutional Neural Network) that has been enhanced and optimized. Images are passed through these convolutional layers to generate a feature map.
- **Neck**: Concatenating and filtering of the extracted features from the Backbone, prominently structured around the SPPF (Spatial Pyramid Pooling Fast) block, enable processing of feature maps with varying sizes while maintaining consistent output dimensions. This significantly enhances the capability to detect features in images.
- **Head**: Give out the predictions base on the loss functions and feature map results. These outputs are bounding boxes coordinates, objects probabilities, classes of the detected objects...

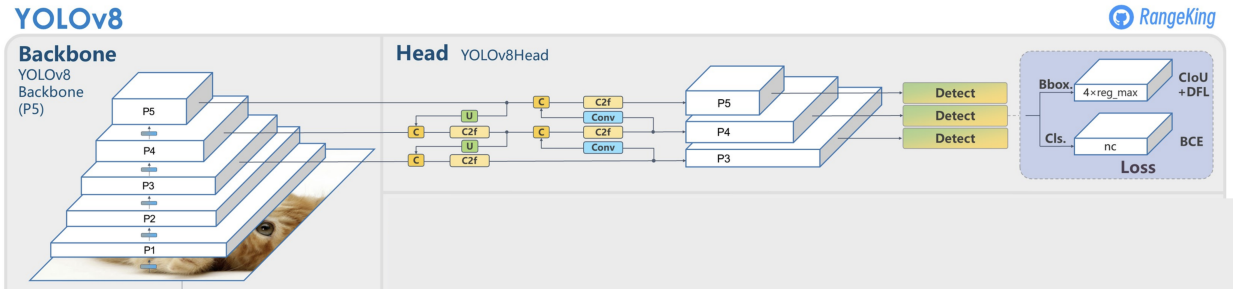

Figure 3.10: YOLOv8 architecture

The YOLOv8 mechanism can be described as follow:

- The images obtained from the camera are pre-processed and converted into RGB images with 3 channels, which serve as the input to the model's first layer. These images are typically pre-processed through OpenCV or other digital image processing tools, standardized in size, and may also be annotated based on specific purposes.
- Afterward, the images are pushed through the Convolution layers of the Backbone to generate a feature map, which represents the image features. Through each layer, the feature map becomes a tensor with varying sizes and more layers. In total, there are 5 convolutional blocks, with 3 blocks concatenated with the neck section to concatenate features.

- The feature layers are stacked sequentially using Concatenate blocks. Alongside, the feature maps are passed through upsample blocks to increase the resolution of the outputs from the SPPF block to match the sizes directly obtained from the convolutional blocks, avoiding dimension mismatches during concatenation.
- Finally, the feature map from the neck section is reshaped to the correct dimensions and fed into the detection blocks, which then make predictions on the feature map based on YOLOv8's loss functions (Bounding box loss, Class loss, etc.).

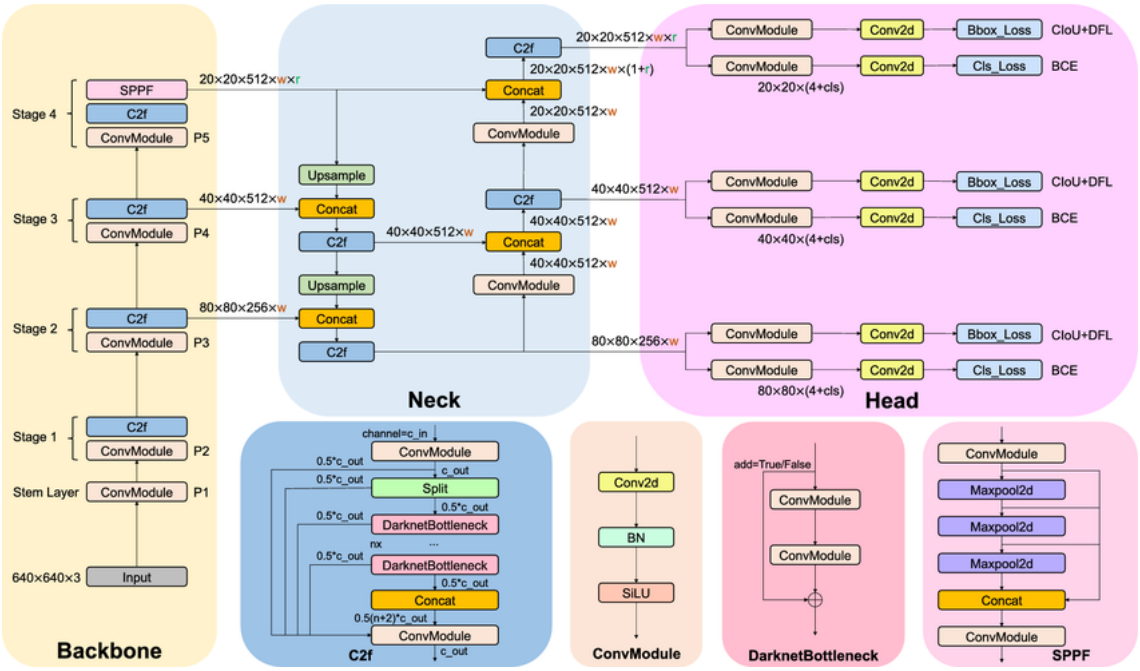

Figure 3.11: YOLOv8 mechanism blocks

3.3 Evaluation

3.3.1 Output

The results return is a list consisting:

$$Y^T = [p_0, (t_x, t_y, t_w, t_h), (p_1, p_2, ..., p_n)] \tag{3.1}$$

With:

- p_0 is the possibility of the object appeared in the bounding box.
- (t_x, t_y, t_w, t_h) are the information of the bounding box, with t_x, t_y are the centerpoints and t_w, t_h are the width and length.
- $(p_1, p_2, ..., p_n)$ are the possibilities of the classes.

3.3.2 Loss function

Loss Function calculate the difference between the predict value and the actual value (ground truth). The goal of training a model is to minimize the loss function value, so that the model can give out the closest prediction. The loss function has 3 properties:

- Classification Loss :Labeling error - the discrepancy arising from incorrect annotations or labels assigned to the data.
- Localization Loss: Parameter deviation of (x, y ,w, h)
- Confidence Loss: Bounding box prediction error - resulting from incorrect predictions of the object's bounding box.

3.3.3 Dataset

In the project, our team deploy a pretrain model. Since YOLOv8 provides various options for pretrain models and sizes depend on the purpose. These models are trained on the COCO dataset [19]. COCO - Common Object in Context is a opensource dataset, developed and maintained by Microsoft, used widely due to the images being common objects in their natural context. With more than 90 classes of objects taken in context, for example human while running, walking, driving or cars on highway, garage, the dataset is also labeled and annotated, making them very convenient for the training. Moreover, COCO is considered a standard dataset, usually used for benchmarking other datasets.

Figure 3.12: COCO Dataset

3.4 Vision module flowchart

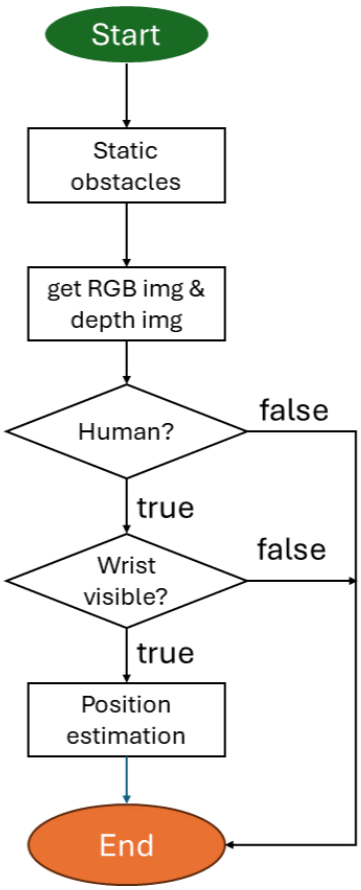

Figure 3.13: Vision module flowchart

This is the flowchart of our vision module, running at the frequency of 25Hz:

- At the start, the camera captures images at a frequency of 25Hz, obtaining both color images and depth images from designated topics output by the camera.
- Upon receiving a color image, the system checks for the presence of a person. If a person is detected, it further checks for the presence of their hand. If these conditions are met, the process proceeds to the next step.
- Upon receiving the depth image, the system calculates the position of the person's wrist in space. Subsequently, it sends these coordinates to a path planning topic.
- After sending the coordinates, if the robot completes its task and stops, the program terminates.

3.5 Coordinates conversion

To determine the coordinates of an obstacle relative to the robot's reference frame, we divide the process into several steps involving coordinate transformations through intermediate coordinate systems.

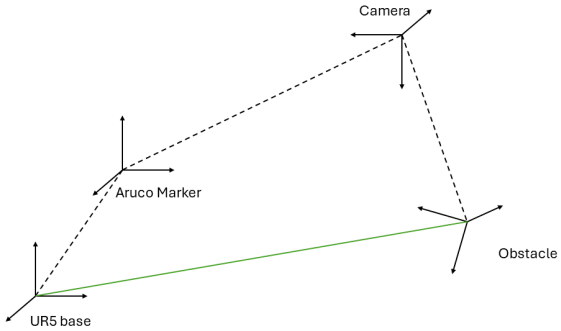

Figure 3.14: Coordinates origin conversion

Robot - Aruco Marker coordinates conversion

Aruco Marker is fixed on the table. The distance between the marker and robot's origin coordinates is predefined. From the camera and the information of the Aruco Marker, we can yield the pose between camera and robot frame.

Aruco Marker - Camera coordinates conversion

First, the camera needs to be calibrated. This can be done using OpenCV commands or by using the Librealsense library provided by the manufacturer. Through this process, we obtain the parameters of the camera's intrinsic matrix.

We need to clearly define the Aruco library, specifying the type and size of the Aruco markers. Then, we generate a set of coefficients for the Aruco markers. When the camera starts running, it captures frames and passes them through the main loop of the program. By using the cv.aruco.detectMarkers command, we obtain information about the corners and IDs of the Aruco markers in the frame. Since the size of the markers is known, we can determine their relative distance and orientation with respect to the camera, allowing us to easily calculate the transformation matrix from the Aruco markers to the camera.

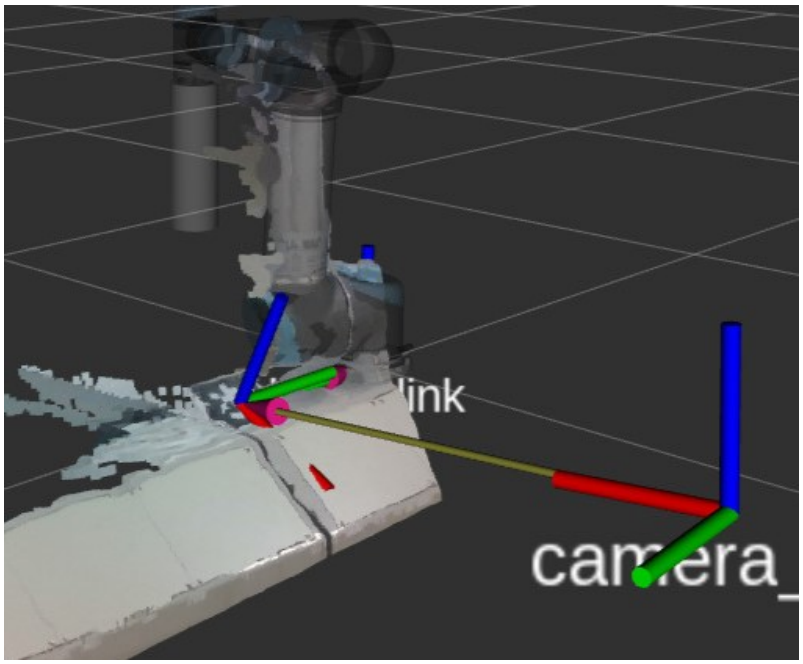

Figure 3.15: Aruco and camera coordinates

The transformation between coordinate systems can then be easily computed using matrix multiplications. In the following section, we will focus on details such as the camera's intrinsic matrix and the relative distance between the camera and the obstacle.

3.5.1 Camera - Human obstacle coordinate conversion

Theoretical basis

The working principle of a camera is similar to a closed box with a single hole for light to enter. The image captured on the sensor is inverted relative to the object, and its size depends on the focal length and the distance between the object and the camera.

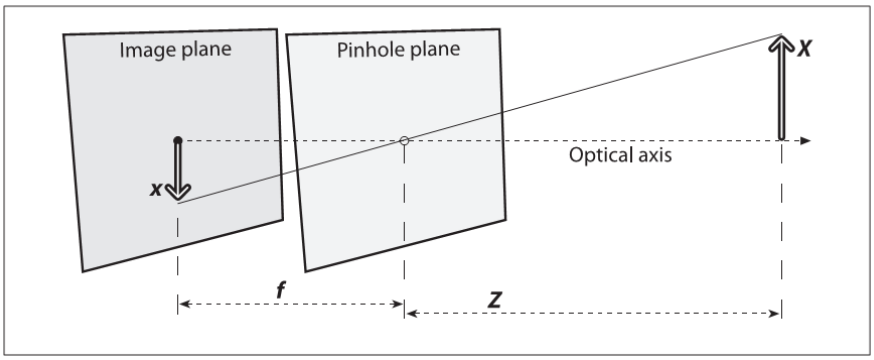

Figure 3.16: Pinhole Camera principles

From this, we can see the correlation between the size of the image and the object, which is expressed through the proportion of two similar triangles:

$$\frac{x}{X} = \frac{f}{Z} \iff X = \frac{Z * x}{f} \tag{3.2}$$

In there:

- **x** is the image's height.
- **X** is the object's height.
- **Z** is the distance from the object plane to the image plan.
- **f** is the focal length.

Based on this, we can determine the relationship between point Q(X,Y,Z) n real space and its image on the image plane as a pixel at position q(x_s,y_s). his relationship is expressed by the following equation:

$$\begin{aligned} x_s &= f_x * \frac{X}{Z} + c_x \\ y_s &= f_y * \frac{Y}{Z} + c_y \end{aligned} \tag{3.3}$$

Trong đó:

- c_x, c_y is the parameters deal with the deviation from the centerpoint of the image to the x and y axis.
- f_x, f_y is the focal length in pixel. (Each pixel on a typical imager is rectangular, so the pixel lengths in x and y are different).
- $c_x, c_y, f_x, f_y, x_s, y_s$ are in pixel.
- X, Y, Z are in meters.

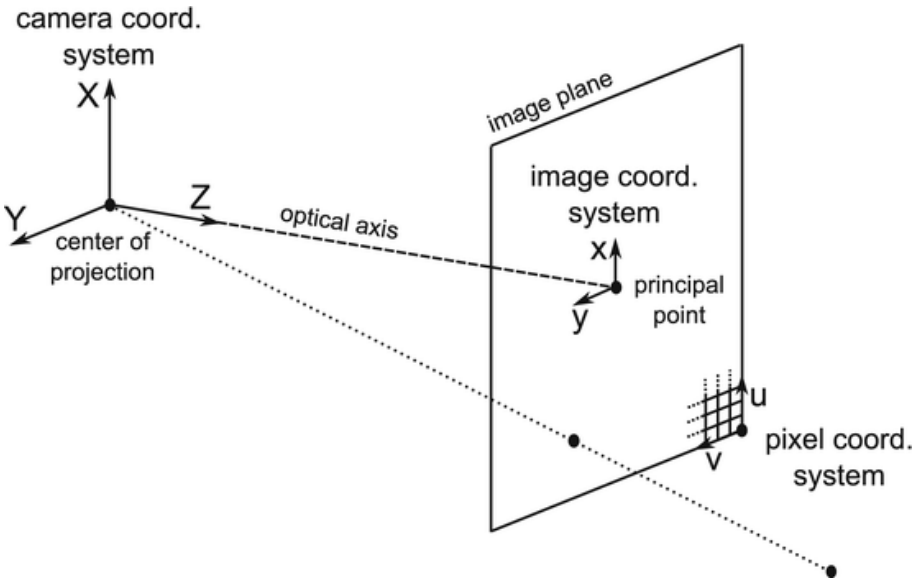

Figure 3.17: Ratio between the object position and its position on the image plane

Camera intrinsic matrix describes the relationship of q on the image plane to the Q on the real space, we have:

$$M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.4}$$

From that, we can calculate:

$$q = M * Q \tag{3.5}$$

3.5.2 Finding camera Intrinsic Matrix

We can get this matrix from the ros topic /camera/depth/camera_info.The intrinsic matrix M will be:

$$M = \begin{bmatrix} 425.4190673828125 & 0 & 431.6041259765625 \\ 0 & 425.419067 & 236.92637634277344 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.6}$$

3.5.3 Predicting human hand using Yolo-pose

Our team has chosen the YOLOv8 model specifically designed for pose detection, which is pre-trained as 'yolov8n-pose.pt'. This is a lightweight model with approximately 3.2 million parameters, providing the fastest inference times. Experimental results from Ultralytics, the developers of YOLOv8, show that the inference speed of 'yolov8n-pose.pt' on an A100 GPU is 1.18ms. This model is well-suited for our system because our goal is to detect a single class (humans), and we need a model that is compact, fast, and has low computational requirements.

Index	Key point
0	Nose
1	Left-eye
2	Right-eye
3	Left-ear
4	Right-ear
5	Left-shoulder
6	Right-shoulder
7	Left-elbow
8	Right-elbow
9	Left-wrist
10	Right-wrist
11	Left-hip
12	Right-hip
13	Left-knee
14	Right-knee
15	Left-ankle
16	Right-ankle

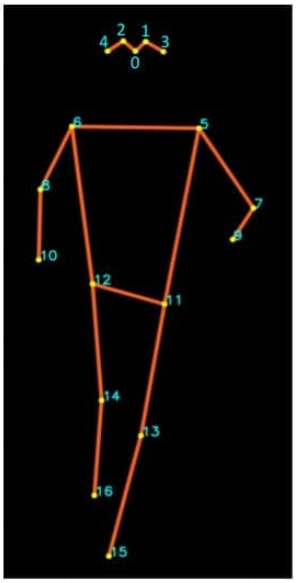

Figure 3.18: Human keypoints detected in YOLOv8

In ROS, our team has developed a node responsible for capturing images from the camera, processing these images, and passing them through the YOLO model. The node then publishes the extracted information (class, bounding box, mask, etc.) to another topic.

The tracker_node.py subscribes to the /camera/image_raw topic of the D435 camera to get the image stream. It creates a publisher YoloResult to publish the processed and recognized images. This is an Image message, generated from the results returned by the yolov8n model. Additionally, YoloResult publishes all results received from yolov8n, including masks and bounding boxes.

Person detection is based on keypoints, which are distinctive points on the body. These points include the eyes, nose, mouth, shoulders, elbows, wrists, and so on.

3.5.4 Human hand detection

In our team's project, humans working alongside the robot will often have their hands occluded or near the robot's movement trajectory. Therefore, we use the YOLOv8 model to track the position of the human wrist in space and send these coordinates to create dynamic obstacles.

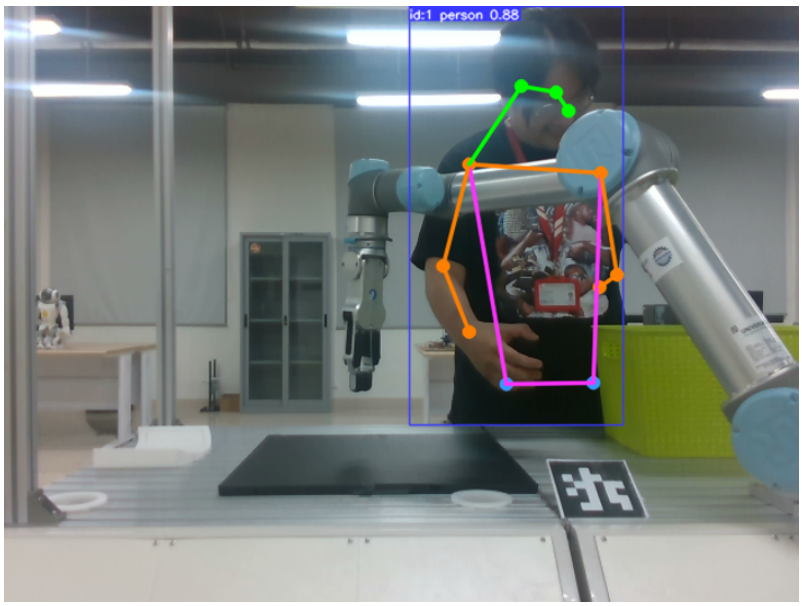

Figure 3.19: Kết quả thu được từ YOLOv8-pose

To increase the system's efficiency, we first check if a person is present in the frame. Only when a person is detected and recognized are the wrist coordinates sent to the planner.

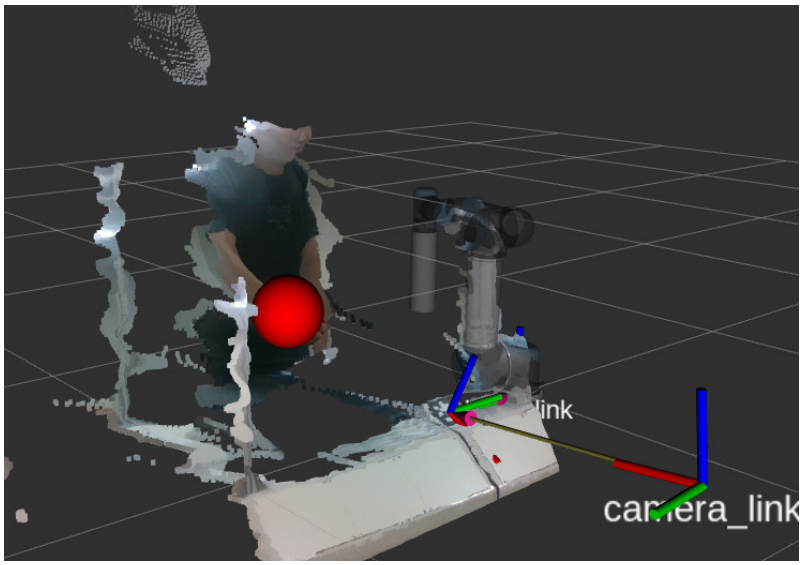

Figure 3.20: Sphere obstacle in planning scene

3.5.5 Create a sphere planning scene

After receiving the wrist position coordinates from the human prediction node, we create an obstacle in the map to facilitate path planning. To conduct the experiment effectively, we will focus solely on the human hand since the hand and the robot typically appear together in the workspace. We will generalize by considering the hand as a simple geometric shape — a sphere. While it is possible to recognize and create obstacles in the map for the entire human body, managing the resultant repulsive force fields would be highly complex and computationally intensive, leading to significant delays in the robot's path execution.

Hence, we create a Primitive object such as Sphere in the PlanningScene - the environment where the robot use to plan its path in MoveIt. According to the research conducted by NASA [4], the average male hand's circumference is 21.8cm.

Gender	Average length	Average breadth	Average circumference
Male	7.6 inches	3.5 inches	8.6 inches
Female	6.8 inches	3.1 inches	7.0 inches

Figure 3.21: Average hand size

In the planning node, we will create a spherical obstacle with a radius of 21.8 cm, centered around the human hand. This sphere is then added to the PlanningScene, and it can be visualized in Rviz as follows:

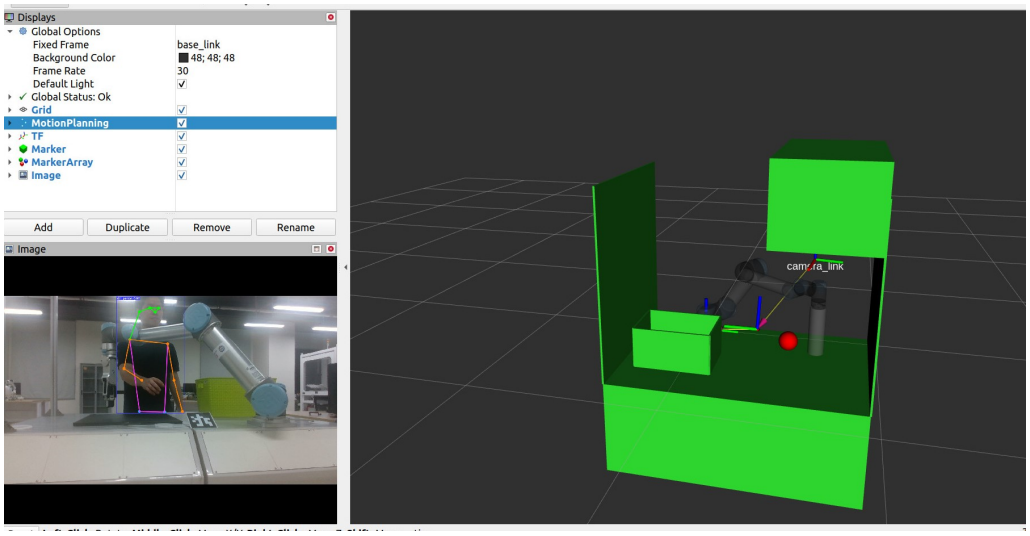

Figure 3.22: PlanningScene of the system, includes the human obstacle

The above results are obtained after processing the environmental information through the vision system. These processed data will then serve as inputs for trajectory planning in the next chapter.

4.1 System components

4.1.1 Robot Operating System

ROS, or Robot Operating System, is an open-source software framework for developing and controlling robots. Developed by Open Robotics, ROS provides a flexible and powerful platform for building, testing, and deploying robotic applications.

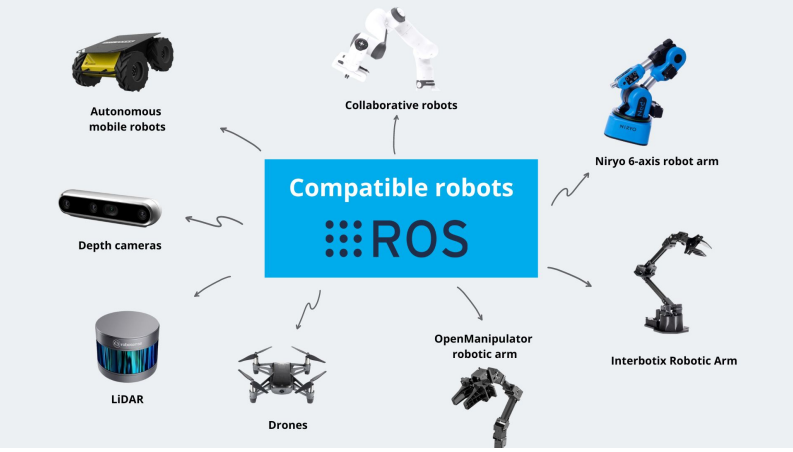

Figure 4.1: ROS Compatibility

Among the versions of ROS, ROS Noetic is not the latest but is quite stable and widely used. ROS Noetic is not only a robust platform for robot development but also has a diverse and active user community. Having a strong community helps users quickly resolve issues and share knowledge, enhancing creativity and efficiency in the development of robots and related applications.

4.1.2 ROS communication and structure

Before talking about each ROS elemnents in our program, we need to understand its file hierarchy. This is very important since it created the unity to the developers' project.

In ROS Noetic, file organization is primarily based on a package structure. A package is a method of organizing software components that pertain to a specific functionality. Overseeing all these functionalities collectively is the Workspace, which provides a working environment for managing and developing the project. ROS also offers the catkin build system to build, manage, and unify resources, thereby creating a systematic and smoothly operating system.

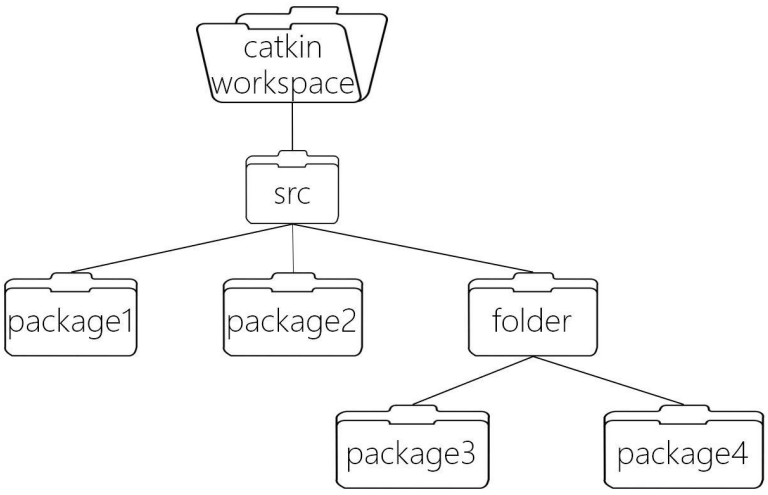

Figure 4.2: ROS file organiztion

After organizing the files, we proceed to the most essential components of the project: the programs that we will write to create our project. In ROS, this component is referred to as a Node. When executing programs, simply put, it is the interaction between two or many Nodes, with this interaction taking the form of packets called Messages. ROS also introduces a special Node that always exists and cannot be replaced, known as the ROS Master (roscore), which manages all information exchanges between the Nodes created by the developer.

However, considering the input-output nature of each Node, ROS delineates several communication protocols. Within the scope of our project, we utilize the two most fundamental communication protocols: ROS Publisher-Subscriber and ROS Service.

Publiser - Subscriber (Pub-Sub)

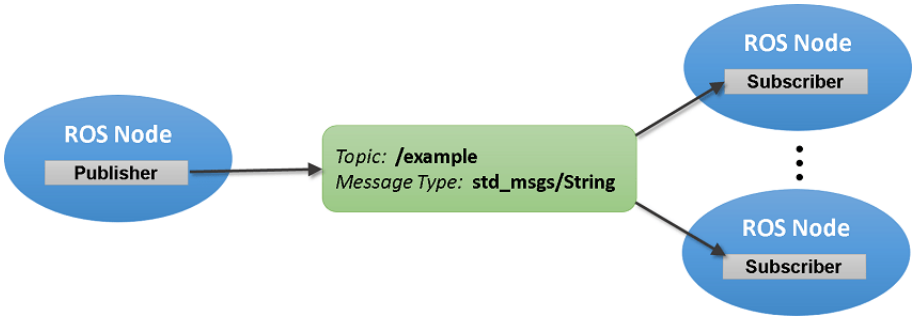

Figure 4.3: Pub - Sub communication

With this type of communicating between the nodes, ROS needs to create a intermediary channel to store the data - called Topic. Nodes are not exactly communicating to each other but rather they communicate with Topic, with information sending nodes are called Publisher and receiving nodes are called Subscriber. A node can be both publisher and subscriber, to on or many Topics. By this, we can easily manage the data flow and understand the program flow. However, this protocol cannot send the information both ways, meaning the data can only has one flow.

Service

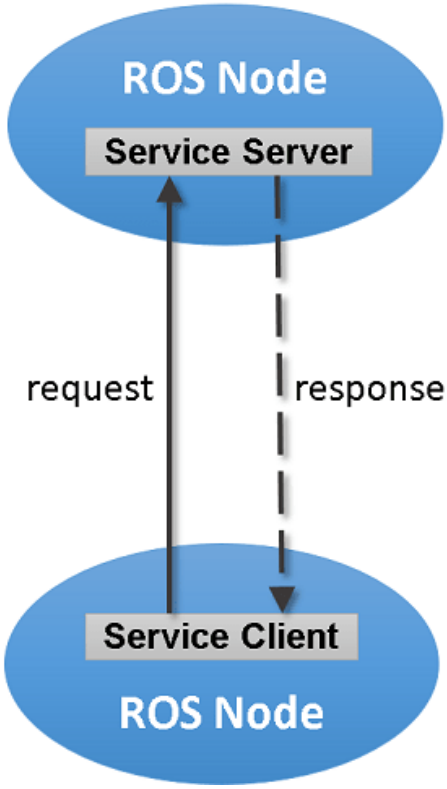

Figure 4.4: Service communication protocol

Unlike Topic, which follows a one-way communication model, Service is implemented using a request/response model. This can be considered a more direct method of communication, where two Nodes exchange information directly without the need for a data storage block. Additionally, the information flows bidirectionally, reducing the complexity of programming and increasing synchronization during responses.

Action

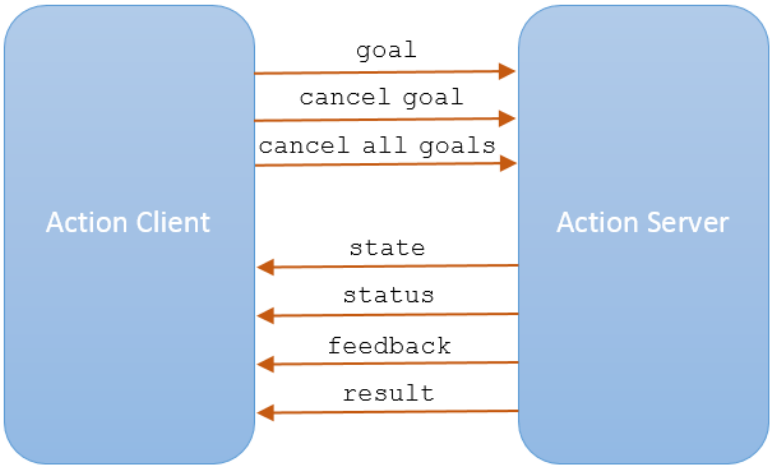

Figure 4.5: Action communication protocol

Additionally, another commonly used communication type is Action, which is often applied in tasks requiring long execution times. Built on top of Topics and Services, Ac-

tions operate similarly to Services but instead of returning a single signal, the feedback from an Action is continuous. Furthermore, Actions can be canceled while they are being executed.

4.1.3 rqt

In ROS, rqt is a graphical tool that aids in visualizing and interacting with data and information from various Nodes within the system. rqt includes numerous extensible plugins, each performing a specific function. These plugins in rqt provide useful tools for monitoring, debugging, and interacting with the components in a ROS system.

In the scope of our project, during the development of algorithms and the use of planning libraries, rqt provided us with a more intuitive understanding of the functionality of each block. This, in turn, helped us select the appropriate planning library for each phase of our process.

4.1.4 Important definitions

Launch file

To execute a program, we have to run multiple nodes in the right sequence, and that should not be done manually in each terminal. That is why we need to write a launch file in the xml format to run all of these nodes in the right order.

URDF

URDF (Unified Robot Description Format) is a format used to describe the structure and geometry of robots in ROS. Using XML format, URDF specifies the relationships between different parts of the robot, including joints, links, and information about their shapes and positions.

URDF helps ROS understand the robot's structure, allowing for visualization and interaction with the robot in a 3D environment. Information such as joints, links, and other parameters are described in detail, helping ROS comprehend how the robot model operates and interacts with its surroundings.

URDF not only introduces the robot's structure but also supports additional features like sensor integration and connection to other devices such as cameras or lidars. This makes URDF a crucial part of robot development and integration within the ROS system, facilitating tasks like accuracy checks, visual representation in Rviz or Gazebo, and managing interactions between the robot and the environment.

4.1.5 ros_control

ROS (Robot Operating System) is an open-source middleware system designed to support the development and control of robots. Within this system, ‘ros_control‘ stands out as an essential package, specializing in the management and deployment of robot control systems.

Optimized for both research environments and industrial applications, ‘ros_control‘ provides a comprehensive solution for the lifecycle of controllers, hardware resource management, and abstraction through hardware interfaces. This package also supports easy integration with third-party solutions for path planning and autonomous navigation execution. This creates a flexible system, enabling robots to adapt efficiently and effectively to a wide range of applications.

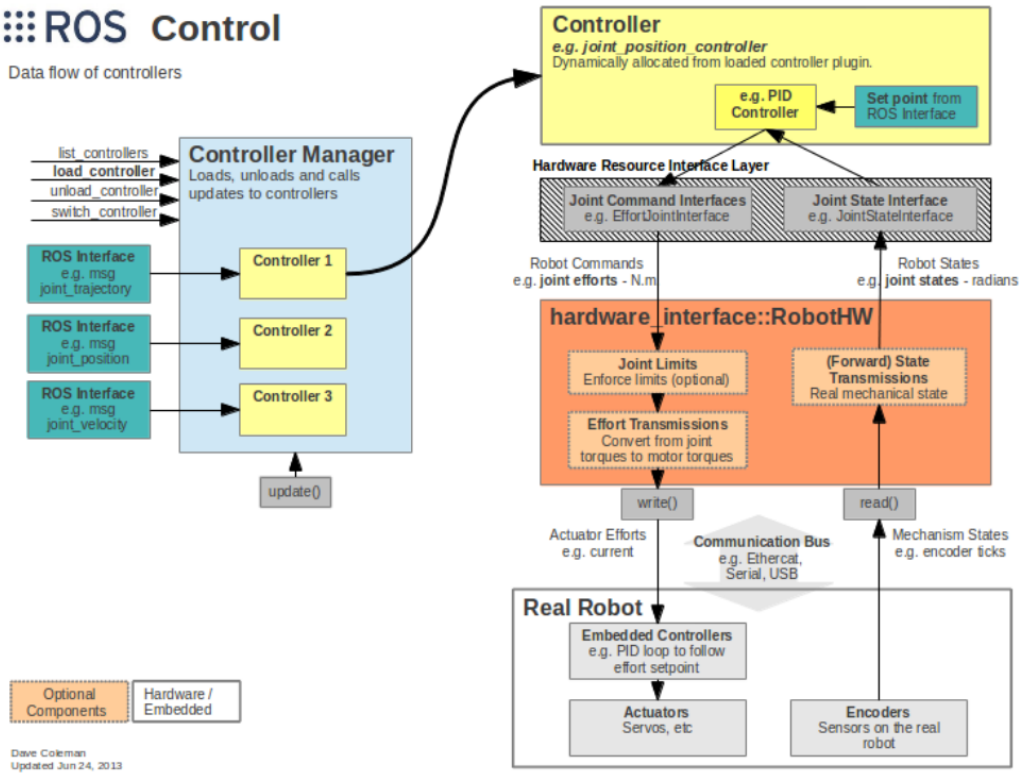

Figure 4.6: Main components of ros_control

Components of ros_control workflow:

- The Hardware Abstraction Layer (HAL): the most important part of ros_control, deployed through hardware_interface::RobotHW, allowing developers to utilize the hardware resources provided by robots, such as electric and hydraulic motors, as well as low-level sensors such as encoders and force/torque sensors.
- Hardware Interface: including state, position, velocity, and effort interfaces, these interfaces are used to abtract the hardware, meaning they provide a way to interact with the harware without directly dealing with the specifics of the hardware implementation, making reading and sending informations of the robot very convenient.
- controller_manager: a component responsible for managing the lifecycle of controllers and hardware resources. It acts as an intermediary between the controllers and the hardware, ensuring that the controllers have access to the necessary resources and handling any conflicts that may arise.
- ros_controllers: pre-built control modules designed to support a wide range of use cases for robotic arms. These pre-designed controllers allow users to avoid building them from scratch, providing ready-to-use solutions for various robotic applications.

4.1.6 rosbag

Rosbag is an important tool in ROS, providing us the ability to record and playback data in the experimenting and testing phase. Rosbag takes and stores all information from sensors and ROS nodes, allowing to do surveys on the data, drawing graphs and picking the right parameters. It also helps in real life situation, in debugging, testing and developing robot efficiently.

After each testing sessions, we changed the parameters and used rosbag to collect the data in order to visualize it through graphs.

4.1.7 matplotlib

Matplotlib is a library for visualizing data in Python. With it, developers can create map, graphs and charts easily, making visualizing data accessible. The library also provides many types of graphs, from pie, column to scatter, histogram... In the project, after we collect the data, we can use matplotlib functions to visualize the data to graphs, and evaluate the project's result.

Figure 4.7: matplotlib library

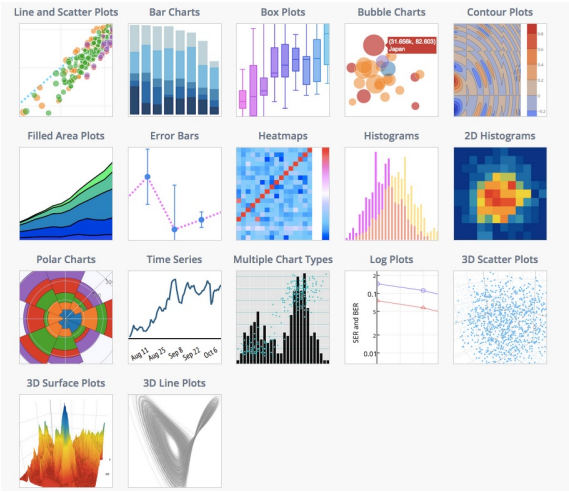

Figure 4.8: Types of plots in matplotlib

4.1.8 UR ROS driver

The ROS-Industrial package developed for Universal Robots (UR) facilitates seamless integration of UR robots into the ROS environment. These packages typically provide drivers, nodes, and other functionalities to simplify control and interaction with UR robots.

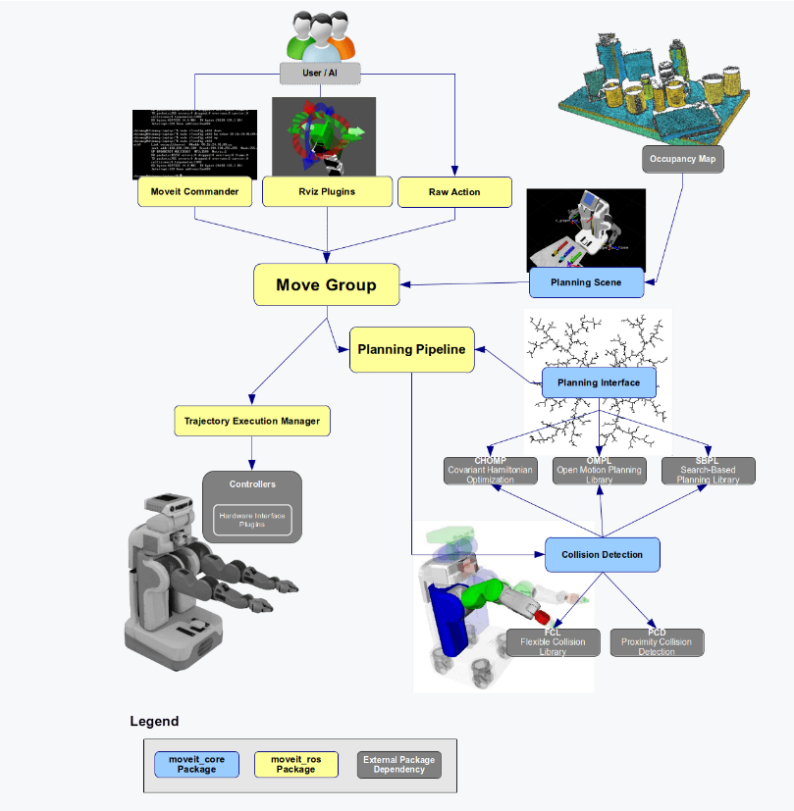

Figure 4.9: MoveIt Pipeline

This driver utilizes ros_control for all control commands. Therefore, it can be used with any position-based controllers available within ros_control or built upon the position_interface framework, as discussed previously.

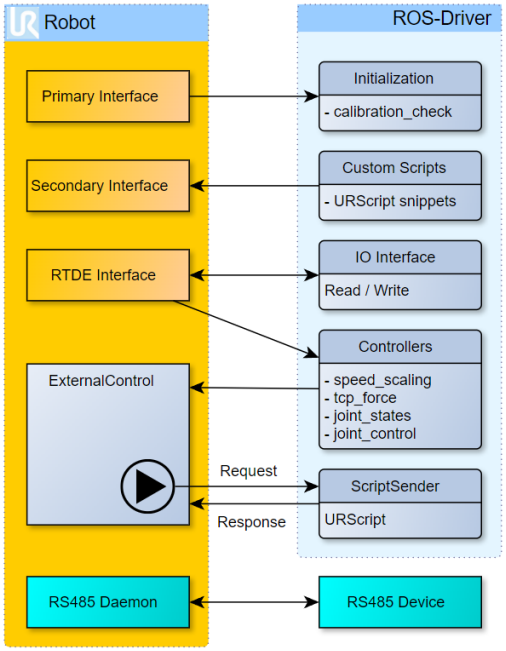

Figure 4.10: Architecture of UR-ros-driver

4.1.9 Flexible Collision Library

FCL - Flexible Collilsion Library is an open source collision checking library. This library can be programmed with C++ language, can be compiled on Linux. This library provides tools for checking collisions and calculating distances between objects, making it very useful in applications such as robot movement planning, simulation, and control. FCL supports many common geometries such as polygons, spheres, and cylinders, and it is integrated into robotic systems such as ROS to improve the performance and reliability of collision detection.

4.1.10 Moveit

MoveIt is a framework for manipulation and has been trusted and used by more than 150 robots.Built on top of ROS, MoveIt can use various ROS tools such as Rviz or URDF, ros_control. The main task of this program is to planning a set of paths for the manipulator to be able to get the last step to the destination, and then execute that path.

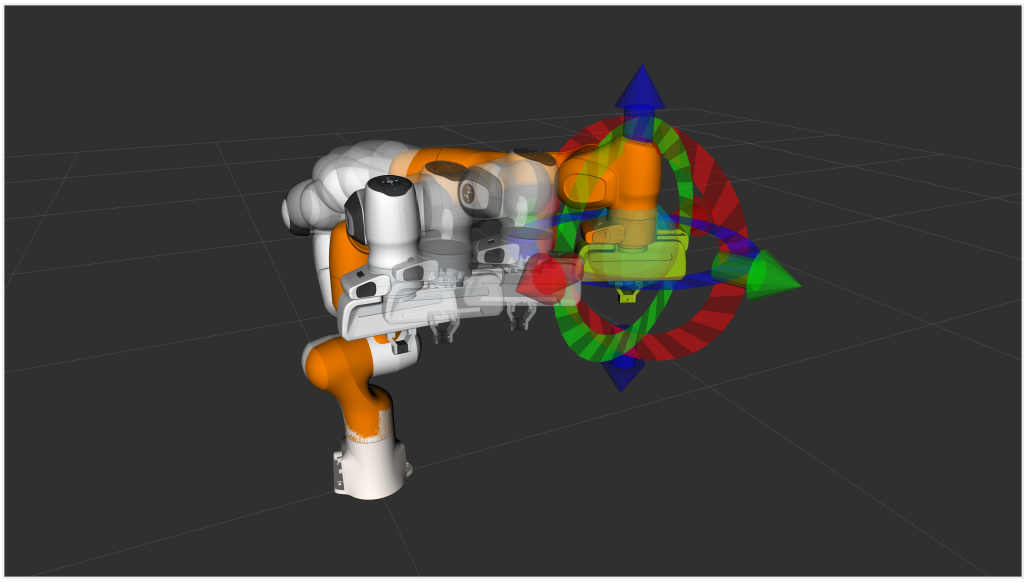

Figure 4.11: Robot manipulator execute a trajectory using MoveIt

The motion planning codes in MoveIt are called **planners**. MoveIt provides various planners, including the use of the OMPL (Open Motion Planning Library), which offers a diverse and well-tested set of motion planning algorithms.

OMPL is an open-source library developed by researchers at Rice University, featuring state-of-the-art sampling-based planning algorithms. It includes pathfinding algorithms such as PRM (Probabilistic Roadmap) and RRT (Rapidly-exploring Random Tree), along with many variations of these algorithms. OMPL is dedicated to path computation and planning purposes and does not handle collision checking or path visualization.

4.1.11 Path planning algorithms overview

Pathfinding algorithms often approach problems in many different ways, including some common and effective methods such as A*, Dijkstra, RRT, Potential Fields or new methods such as using AI and machine learning. or Hybrid - combine methods together. Some characteristics of the methods:

- **A*, Djikstra** are search-based algorithms. The Dijkstra algorithm can be understood as an exhaustive search algorithm. When the environment is divided into grids, the search tree will sequentially connect nodes until it finds the shortest path from the

start point to the destination. The A* algorithm improves upon Dijkstra's algorithm by incorporating a heuristic condition during the search. Typically, after moving from the start point to the next node, A* checks the distance from there to the destination. The node that returns the shortest distance to the destination is selected as the next node. A* minimizes computational resources by not having to search all edges in the search tree.

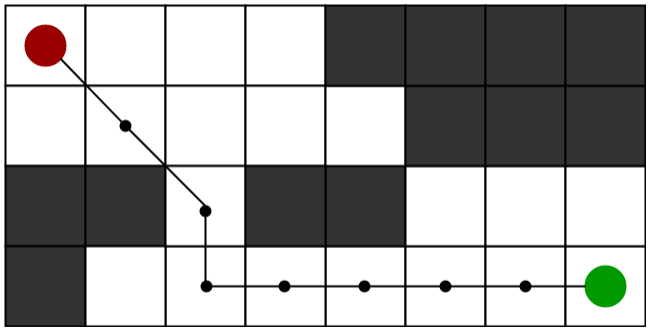

Figure 4.12: A* Algorithm

- **RRT** (Rapidly-exploring Random Tree) is a sample-based algorithm.Initially, we have a start point and an end point, and we randomly sample within the environment. Connect any randomly selected node to the nearest node, and on that connecting path, plan an edge of the tree. This edge must adhere to a certain maximum distance condition – the maximum distance that can be traveled between two nodes. If there are obstacles between these two nodes, no edge will be created. By setting this condition, we can avoid moving too far from the destination or into unsafe areas. This process is repeated, creating randomly planned branches of the tree that continuously spread into unexplored regions. This algorithm is useful when we need to find a feasible path that does not necessarily have to be the optimal path.

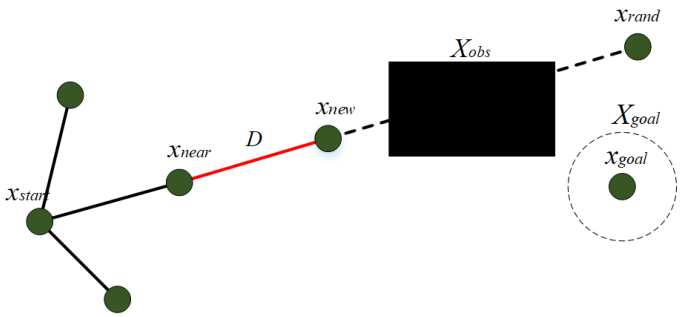

Figure 4.13: RRT Algorithm

- **APF** (Adaptive Potential Fields) takes inspiration from force fields such as gravity or electromagnectics, by making repulsive and attractive force fields to guide the path of the robot. Although it is not complex and computational heavy, APF usually encounter the problem of Local Minima - robot locked in a position due to the equal forces of push and pull.

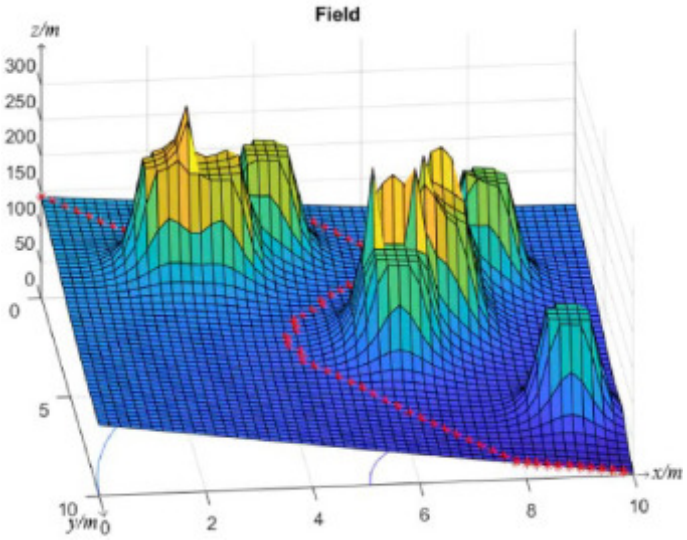

Figure 4.14: Artificial attractive and repulsive fields

Considering the suitability of the algorithms to our project, we decided to experiment RRT and APF to improve and overcome their downsides.

4.2 RRT algorithm in manipulator

Rapid-exploring random tree - RRT [4.13] is designed to find solutions in high-dimensional, non-convex spaces by constructing a search tree that fills the space. This tree is continuously built by random sampling, ensuring it always expands into unexplored regions of the space. RRT easily handles path planning problems in obstacle-laden environments and is widely applied in Robotics.

4.2.1 RRT variations

Since RRT is not inherently an optimal algorithm, researchers introduced the RRT* algorithm to optimize it over time. RRT* generates new nodes randomly similar to RRT. The key difference in RRT* is that instead of connecting edges to the closest neighbor, the algorithm introduces a neighborhood region parameter - a circular area encompassing nearby nodes. Within this region, the algorithm calculates and connects the edge to the point where the distance from the start to the newly created node is shortest.

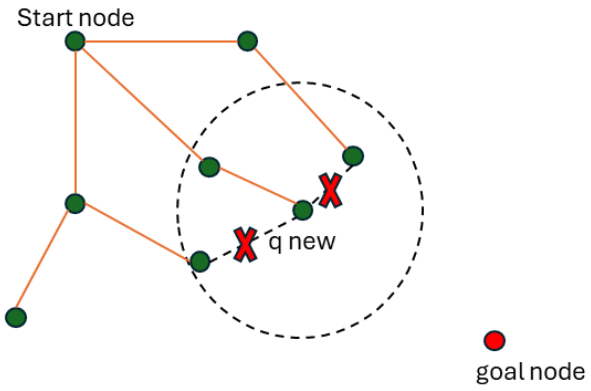

Figure 4.15: RRT* algorithm connection

In the project, we use the RRTConnect algorithm in the OMPL library. This is the

bi-directional RRT algorithm, instead of growing the tree from the start point, the tree will grow from both the start and the goal point. After each node, the two trees have the tendency to get closer and connect to the other one through the newly created node and the closest node from the other tree. RRTConnect is usually giving out a faster and better results than RRT.

4.2.2 RRTConnect results

After planning using RRTConnect, we got the following path. Because this is not an optimized path, each RRT iteration will give out a different though still feasible path.

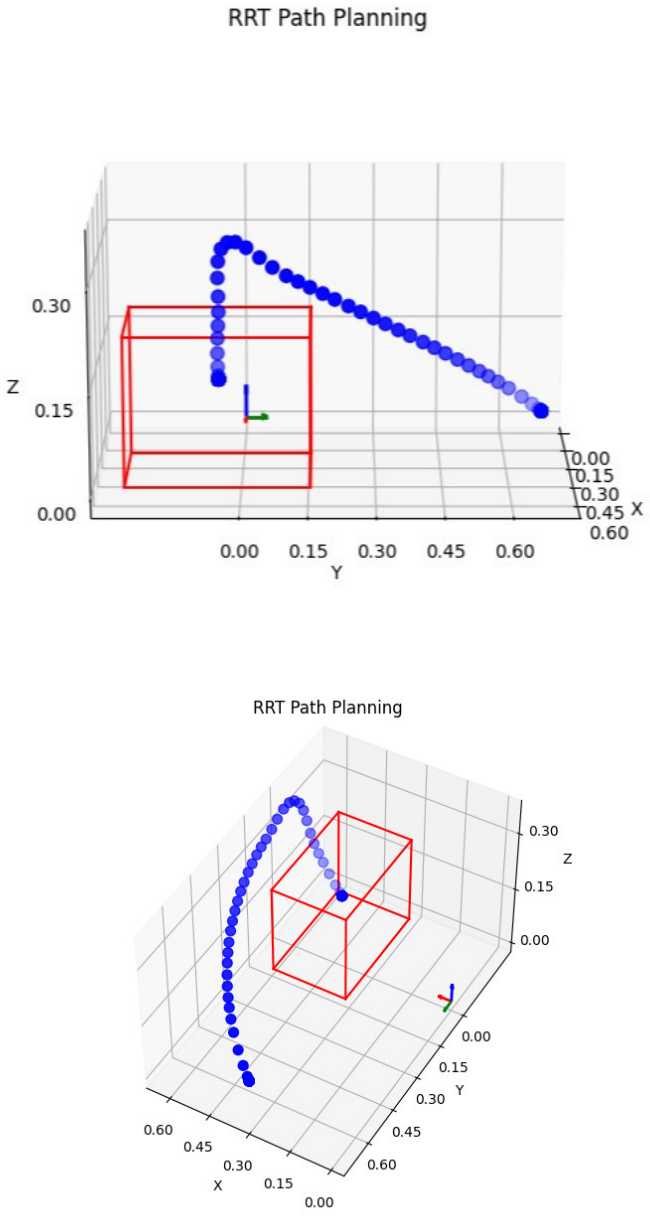

Figure 4.16: RRTConnect path

4.3 Artificial Potential Field (APF)

4.3.1 Theoretical basis

The artificial potential field method is widely used in autonomous robots and was proposed for the obstacle avoidance problem in robot arms by [12]. Assuming the robot

is a point, the idea of this problem is to create an attractive force field that pulls the robot towards the target point in the workspace while generating repulsive force fields at obstacles in the workspace. Path planning in this way is computationally efficient because the robot does not need to compute and search for valid paths.

Assume there exists an obstacle *O* in the workspace. If the robot's final configuration is q_d, the initial configuration q_0 can move to the target by placing it in an artificial force field U_{art}:

$$U_{art}(q) = U_{x_d}(q) + U_O(q) \tag{4.1}$$

Where:

- $U_{att}(q)$ is the attractive potential field, directs the robot towards the goal.
- $U_{rep}(q)$ is the repulsive potential field, steers the robot away from obstacles.

Considering the attractive field, the field has a local minimum set at the goal configuration of the robot. This creates a force "well", tending to pull the robot to that pose. The author proposed:

Conic Well

With the ideal creating a cone-shaped field, the Conic well can be described as:

$$U_{att}(q) = k|q - q_0| \tag{4.2}$$

This is a linear function, proportional to the distance between the robot configuration to the goal configuration. However, taking the gradient of the fields, we can see that this is not a continous function, at the minimum value - the goal.

Parabolic Well

To solvting the continuity problem, researchers proposed using a parabolic function. The new attractive field will be:

$$U_{att}(q) = \frac{1}{2}k(q - q_d)^2 \tag{4.3}$$

This function will fix the mentioned problem, but it does not have an upper bound, leading to its quadratic growth to infinity when the distance between robot and the goal configuration increase.

Therefore, combining two idea will give the most reasonable result, we can write the attractive field as:

$$U_{att}(q) = \begin{cases} \frac{1}{2}k(q - q_d)^2 & ; |q - q_0| \leq d \\ dk|q - q_0| - \frac{1}{2}kd^2 & ; |q - q_0| > d \end{cases} \tag{4.4}$$

The attract force is the result of the field at q_d, proportional to the paramenter k_p, can be written as:

$$F_{att}(q) = -\nabla U_{att}(q) \tag{4.5}$$

The repulsive force is generated by the repulsive field associated with the obstacle *O*, selected such that the field $U_O(q)$, or U_rep is a continuous and differentiable function ensuring $U_{art}(q)$ is zero when q = q_d. In [12], the author proposed the formula:

$$U_{\text{rep}}(x) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho} - \frac{1}{\rho_0}\right)^2 & ; \rho \leq \rho_0 \\ 0 & ; \rho > \rho_0 \end{cases} \tag{4.6}$$

Where :

- ρ_0 is the set distance of effect of the field.
- ρ is the closest distance to the obstacle *O*.
- η is the proportional parameter.

Here, due to the complexity of calculating the position of obstacles in the joint coordinate system, the distances to the objects are calculated and defined in the common coordinate system. Therefore, the repulsive force obtained from the gradient of the function $U_{\text{rep}}(x)$ will be represented in the common coordinate system.

The condition of ρ_0 is set to ensure that the robot will be pushed away when it is within a certain region around the obstacle. In that case, the repulsive force $F_{rep}(q)$ is calculated as:

$$F_{rep}(x) = \begin{cases} \frac{1}{2}\eta(\frac{1}{\rho} - \frac{1}{\rho_0})\frac{1}{\rho^2}\nabla\rho & ; \rho \leq \rho_0 \\ 0 & ; \rho > \rho_0 \end{cases} \tag{4.7}$$

Where $\nabla\rho$ is the gradient of the vector from the affected point to the obstacle *O*.

$$\nabla\rho = \frac{\partial\rho}{\partial x} = \begin{bmatrix} \frac{\partial\rho}{\partial x} & \frac{\partial\rho}{\partial y} & \frac{\partial\rho}{\partial z} \end{bmatrix}^{\mathrm{T}}. \tag{4.8}$$

In the general case of robotic arms, the attractive and repulsive forces need to be calculated based on the distance from the robot to the obstacles and the target. Since accurately calculating the distance from points on the robot, represented in the joint coordinate system, to the points of obstacles and targets in the joint coordinate system (C-space) is not easy, researchers often propose selecting a fixed set of points on each segment of the robot as the locations for applying the attractive and repulsive forces. These points are called control points, and in this study, we decided to choose these points as the joints of the UR5 robot.

However, when applied to robots or specifically robotic arms, this theory encounters several drawbacks:

- Local Minima: Combining multiple force fields will create a local minima. This is not the goal configuration, and has the tendecy to pull the robot to a locked position due to the sum of the forces' vector at the point is zero.

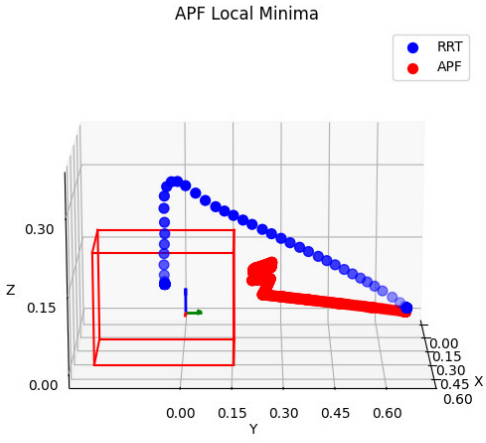

Figure 4.17: Local Minimum in path planning

- Robot can not reach the goal due to that path is in the area of effect of the repulsive fields, also known as Goals Non-Reachable with Obstacles Nearby (GNRON [7].

4.3.2 Adaptive APF

To address GNRON, [29] proposed a change in the repulsive field formula:

$$U_{\text{rep}}(q) = \begin{cases} \frac{1}{2}\eta(\frac{1}{\rho} - \frac{1}{\rho_0})^2 \frac{p_g}{1+p_g} & ; \rho \leq \rho_0 \\ 0 & ; \rho > \rho_0 \end{cases} \tag{4.9}$$

Trong đó:

- p_g is the position from control point to the goal

In 4.9, when the distance between joints to the target point is larger, $\frac{p_g}{1+p_g}$ approaches 1, causing the attraction force to have a value similar to the classic AAPF case. On the other hand, when the distance between the robot and the target point is small, which corresponds to $2\frac{p_g}{1+p_g}$ approaching approximately p_g, the attractive force acting on the robot will no longer hinder the attractive field generated at the target point.

In AAPF, the repulsive force can be split into 2 components:

$$F_{rep} = -\nabla U_{\text{rep}}(x) = \begin{cases} F_{rep1} n_{OR} + F_{rep2} n_{RG} & ; \rho \leq \rho_0 \\ 0 & ; \rho > \rho_0 \end{cases} \tag{4.10}$$

Where:

$$F_{\text{rep1}} = \eta \left(\frac{1}{\rho} - \frac{1}{\rho_0}\right) \frac{\rho_g^n}{\rho^2(1+\rho_g^n)} \tag{4.11}$$

$$F_{\text{rep2}} = \frac{n}{2}\eta \left(\frac{1}{\rho} - \frac{1}{\rho_0}\right)^2 \frac{\rho_g^{n-1}}{(1+\rho_g^n)^2} \tag{4.12}$$

$$n_{OR} = \nabla \rho \tag{4.13}$$

$$n_{RG} = \nabla \rho_g \tag{4.14}$$

F_{rep1} will be the component that helps the robot to avoid the obstacle and F_{rep2} is the component to guide the robot to the goal.

4.3.3 Local Minima Problem

Due to the attractive and repulsive forces in space, coupled with the presence of moving obstacles like human arms, predicting the positions of local minima or 'traps' is very challenging. In our experimental setup, the robot gets stuck because it cannot escape from a local minimum, where the repulsive and attractive force vectors are approximately equal and opposite in direction. This causes jerky movements in the robot's path when it falls into such a scenario.

To address the issue of local minima, our team proposes a new control approach that combines RRT and AAPF. In this approach, AAPF acts as the local planning module to solve obstacle avoidance problems. By doing so, the region that AAPF needs to handle between the starting point and the goal point is significantly reduced. Moreover, whenever the robot encounters a local minimum, the global planning module RRT will propose a new goal point to alter the attraction force field, helping the robot escape from local minima.

Using AAPF yields satisfactory results when the robot's starting and goal points are not obstructed by obstacles, allowing the attraction force vector to guide the robot towards the goal while the repulsive forces only serve to steer.

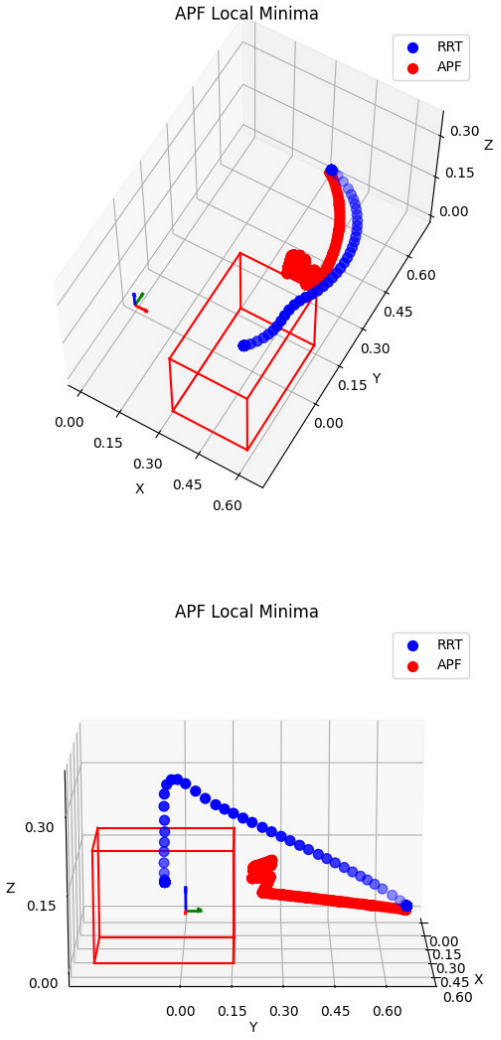

Figure 4.18: Robot stucked in local minima

4.3.4 Torques/Forces applying on the robot

Base on the theory, we can calculate the forces that these fields applied on the robot body, so that we can use the dynamic equations to find the control parameter.

Considering the robot and time stops at every step we calculate the field, we can calculate the forces/torques applied on the robot as:

$$\tau(q) = \tau_{att}(q) + \tau_{rep}(q) \quad = -\nabla U_a tt(q) + {J_T}^T F_{rep} \tag{4.15}$$

Với ${J_T}^T$ là ma trận Jacobi tịnh tiến của UR5.

4.3.5 Gradient Descent

In Machine Learning specifically, and Optimization in general, we often need to find the minimum (or sometimes maximum) value of a given function. For functions that are multi-dimensional, non-linear, or non-continuous, this task can be very complex. Therefore, Gradient Descent is commonly used to find local minima, which can be considered solutions to the problem.

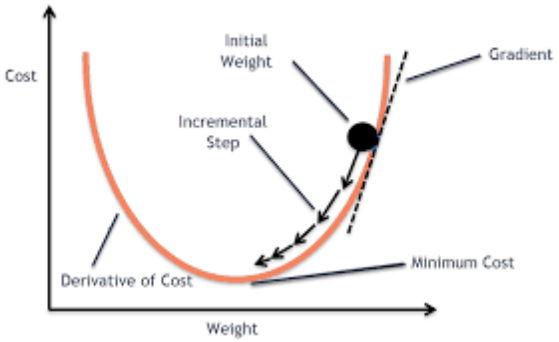

Figure 4.19: Gradient Descent Example

The idea of Gradient Descent can be explained simply as taking small steps (controlled by the learning rate) in the direction of the negative gradient (hence the term 'descent') iteratively through each loop. The algorithm's task is to converge to the nearest possible minimum of the function. Due to the proportional dependence on the learning rate parameter, applying Gradient Descent can encounter several issues.

The path planning problem using the AAPF method involves seeking the global minimum of the potential field generated by obstacles and the goal. After determining initial and final configurations through inverse kinematics steps, we establish the attractive potential field. Using a camera, obstacles are detected to create corresponding repulsive fields, enabling the calculation of the total moments acting on the robot in joint coordinates. Here, we employ the gradient descent method to control the robot towards the desired configuration.

Algorithm 1 Gradient Descent

$q_{\text{start}} = q(0)$
$i = 0$
while $\nabla U(q_i) > 0$ and $i \leq i_{\max}$ **do**
  $q(i+1) = q(i) + \alpha \left(\frac{\tau_{\text{att}}(q_i) + \tau_{\text{rep}}(q_i)}{\|\tau_{\text{att}}(q_i) + \tau_{\text{rep}}(q_i)\|} \right)$
  $i = i + 1$
end while
return $q = (q_0, q_1, ..., q_n)$

Ưhere α is the step of the gradient descent.

4.3.6 Path planning with AAPF

Below are the results when our team implemented path planning using the AAPF algorithm, demonstrating the instability observed when the robot fails to complete its movement path due to falling into local minima.

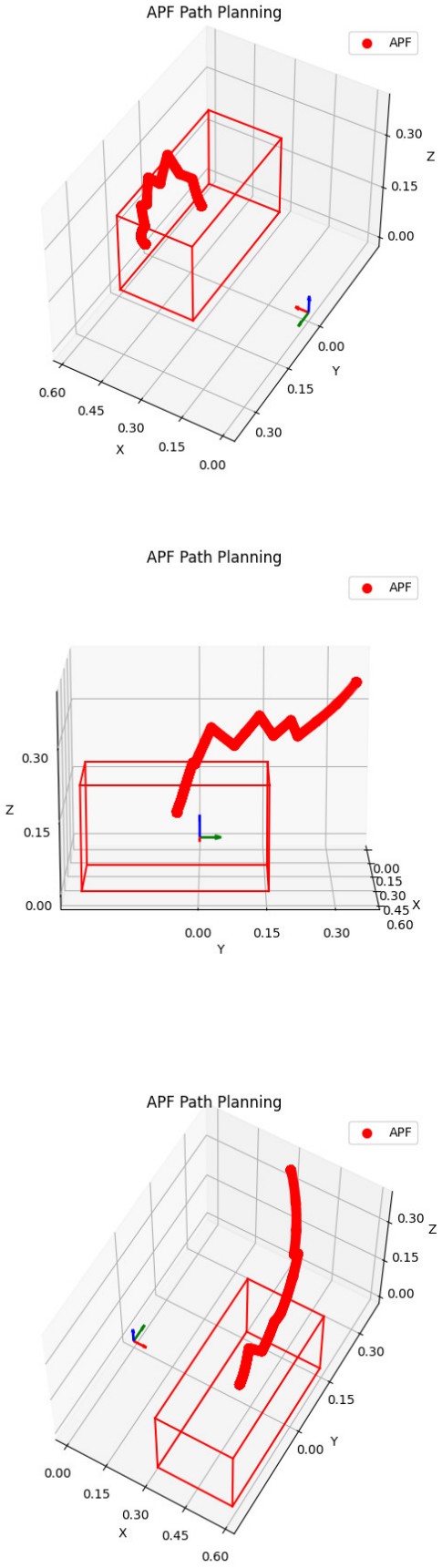

Figure 4.20: Path planned by AAPF

Since the drawbacks and the advantages of these 2 algorithm, we proposed a hybrid

planner as follow.

4.4 Hybrid planner of RRT and AAPF

4.4.1 Hybrid Planner flowchart

Based on these algorithms, our team proposes using the RRT algorithm as the global planning framework, leveraging sensor signals and sensor information to generate a path from the start to the goal. Subsequently, the AAPF method is applied between consecutive nodes, tasked with guiding the robot through sets of waypoints along the path. The use of AAPF allows for handling dynamic obstacles such as human arms in the working environment.

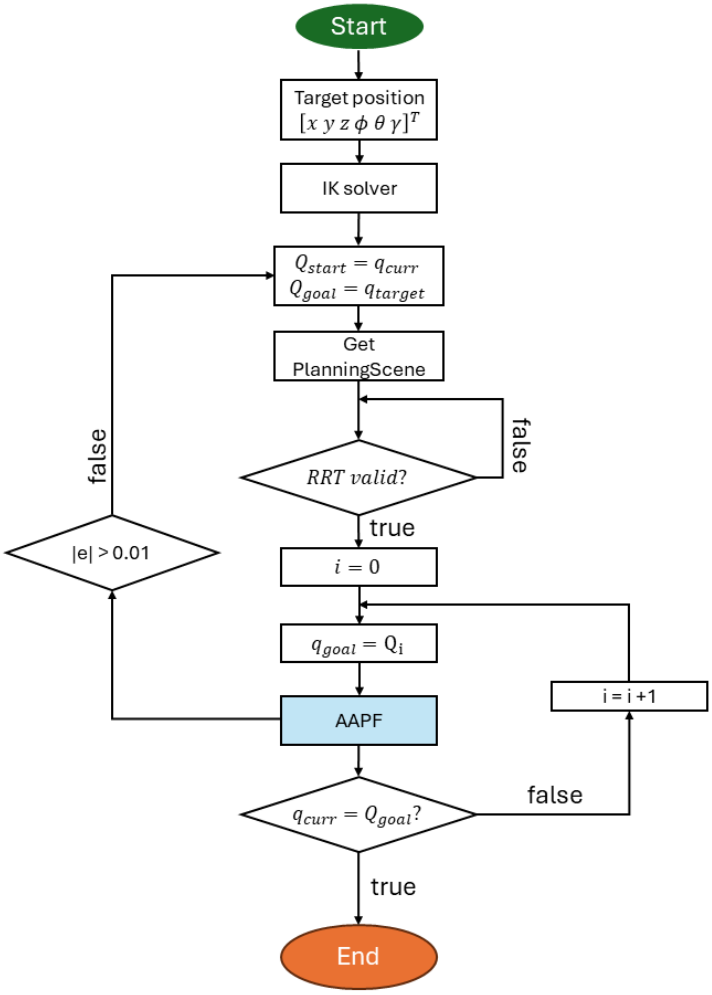

Figure 4.21: Planner flowchart

The system overview could be seen in 4.21.In this planner, the RRT serves as the global planner, generating a set of waypoints. These waypoints are used as reference paths, gradually fed into the AAPF planner to create forces guiding the robot towards the final destination. Specifically:

- As the system starts, the goal point position in the base coordinates system, represented as $[x, y, z, \theta, \phi, \gamma]$. This will be transformed to the joint coordinates system through an Inverse kinematics solver. We will have the start and goal configuration $Q_{start}t$ and Q_{goal}.
- The program will update the PlanningScene, checking whether if there is human hand or not. In case the hand was detected, its position will be updated in the Plan-

ningScene.

- After that, two configurations are used by the RRT algorithm to generate reference trajectories. This is recalculated if the PlanningScene is updated with the coordinates of the human hand, at which point a new set of waypoints for the path will be generated as a new reference.
- With the set of reference paths, we sequentially input two consecutive points into the AAPF. The robot is gradually pulled from configuration q_i to q_{i+1} by creating attractive forces at q_{i+1} and assigning repulsive forces to obstacles in the PlanningScene. Here, $|e|$ represents the difference between q_i and q_{i+1}, The AAPF block checks the condition of |e|, if it is smaller than a predefined threshold value, the program sets the current robot configuration as Q_start. This prompts the RRT to generate a new set of reference paths, thereby minimizing and eliminating jerky movements between closely spaced configurations.
- Finally, the robot will check its configuration to see if it is in the final goal or not. If it is, the planning program will end.

Realizing the planning procedure between 2 points by attractive and repulsive forces will be very difficult to spot and avoid local minima when unknown obstacles appears (human hands). This will likely guide the robot to traps, or create jerky path when calculating forces between these points. Our planner takes this into consideration, as we set predefined constraints, checking and if necessary creating a new set of reference points.

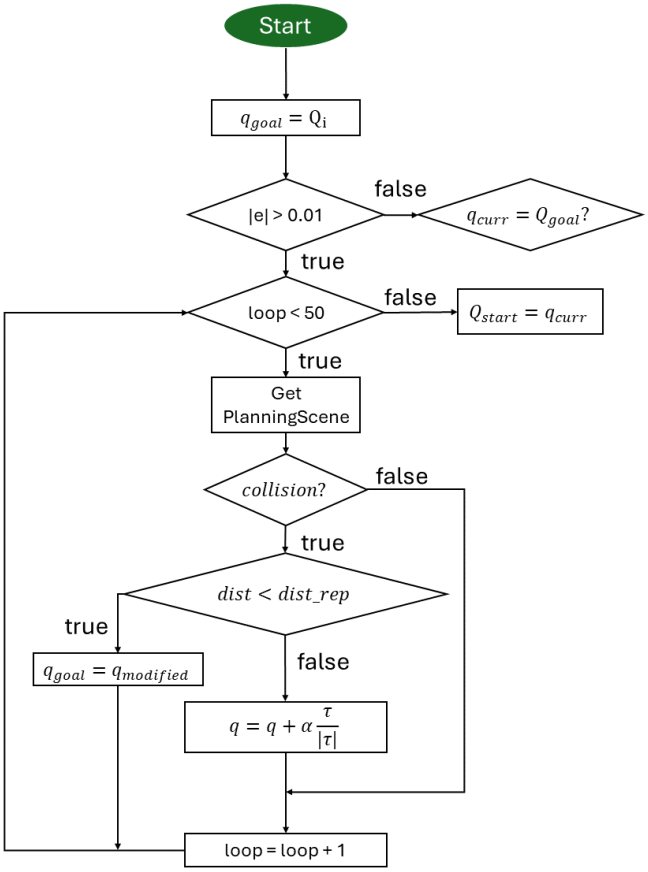

Figure 4.22: AAPF flowchart

The AAPF is the local planner, solving the planning problem when human hand is detected, and can be described in 4.22:

- After new goal point Q_i in the reference point are fed into the block, we calculate |e| and check its constraint, if it is bigger than 0.01 (rad), the program will proceed.
- AAPF block's main task is to create the path from q_i to q_{i+1}. At the main loop, we check for collision. In case there is none and the robot configuration are still under the influence of potential fields, we will use gradient descent to find the waypoints.
- To avoid the case of attractive and repulsive forces applied on the robot are equal, leading to the path oscillation and never reaches the goal (GNRON), we will check the constraint of distance between robot to the obstacle. The oscillation happens due to the fact that goal point q_{i+1} is too close to the obstacle. The team proposes that when this distance falls below a certain threshold, a new reference point $q_{modified}$ will be created. This point is created by moving q_{i+1} along the direction of the repulsive forces until it escape the repulsive field's influence. Then the planning will be calculated between q_i and $q_{modified}$.
- By doing this, we can minimize the jerkiness of the path, as well as following the reference path without deviating too much when encountering human hand.

Where:

- $Q_{start}t$, Q_{goal} ... are the robot configuration references (starting and goal configurations).
- q_i, q_{i+1} are the reference fed into the AAPF.
- $|e|$ is the difference between q_i and q_{i+1}.
- $dist$ is the closest distance between the control points to the obstacle.
- $dist_rep$ is the range of influence of pnstacle repulsive field.

4.4.2 AAPF cases

Goal point is not under repulsive field's influence

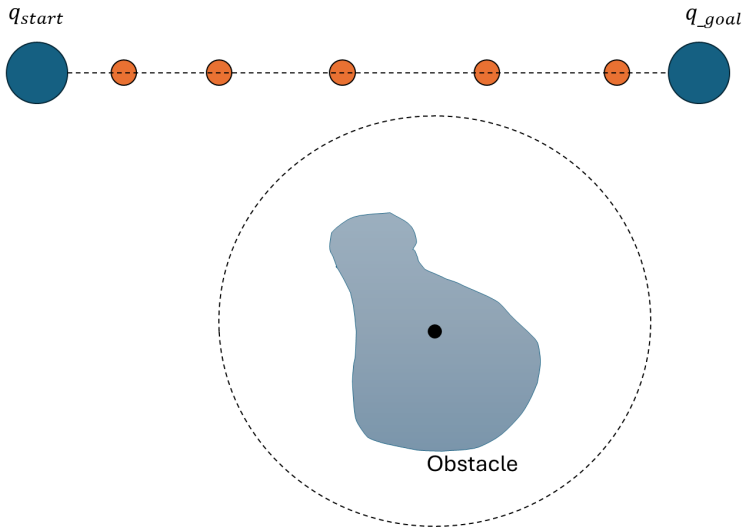

Figure 4.23: Path planned when goal point is not under repulsive field's influence

When the 2 reference points q_i, q_{i+1} are fed in, in the case of the obstacle in PlanningScene creates a repulsive field that does not affect the robot configuration, the path created will be guide straight to the goal by the potential fields. This will be a smooth path from q_i to q_{i+1}.

The constraint of |e| < 0.01 is to check whether the robot has reach its final goal configuration or not, so that it could continue feeding in another pair of start- goal reference point. The planner only stops when the robot has finished planning to the final reference goal Q_{goal}.

Goal point is under repulsive field's influence

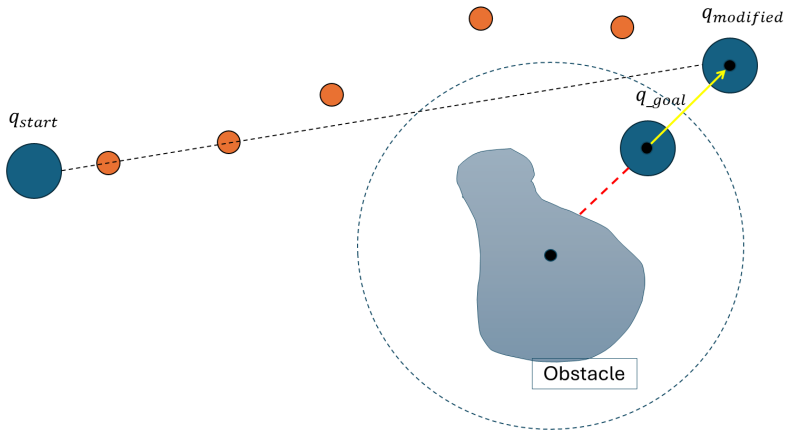

Figure 4.24: Path planned when goal point is under repulsive field's influence

In this case, we consider the scenario where the destination point q_{goal} lies within the influence region of the repulsive force field. This occurs possibly because the points generated by RRT are too close to known obstacles, or due to the sudden appearance of a human hand obstacle.

This is the reason for the jerky motion of the path, as when the robot approaches the destination within the force field's range, the repulsive force increases while the attractive force decreases. This continuously pushes the robot away until it manages to escape this influence zone.

We proposed to create a new reference point $q_{modified}$ outside the influence area. We will use the repulsive force vector generated by the obstacle. Considering the direction vector from the nearest point on the obstacle to the robot's control points, we will translate these points outside the repulsive force's influence region along this vector.

The new reference point is shifted based on the repulsive force, which can be calculated using the formula:

$$q(i+1) = q(i) + \alpha \left(\frac{\tau_{\text{rep}}(q_i)}{\|\tau_{\text{rep}}(q_i)\|} \right) \tag{4.16}$$

Gradient descent will be calculated until the distance from the robot to the goal is bigger than a defined value, meaning the robot has escaped the obstacle repulsive field.

Goal point is occluded by the obstacle

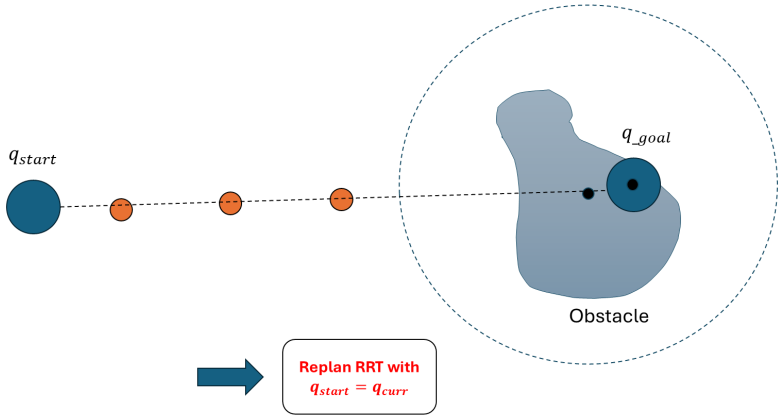

Figure 4.25: Path planning when goal point is occluded by the obstacle

This is the scenario where obstacles obscure the goal point. This situation occurs when a human arm appears within the workspace, obstructing the q_{goal} that the AAPF block is moving towards.

This will be checked using a collision check block. In this case, the robot will pause and increase the loop without performing any calculations. This is to wait and see if the human arm remains in place or continues to move. If the arm moves, it will move away from the reference point's position. Conversely, if the arm remains stationary, we consider it a known static obstacle and proceed to recalculate the RRT.

In this case, the current position of the robot configuration $q_{current}$ will be chosen as the starting point for the reference path, while the goal point of the reference remains unchanged. Recalculating the RRT will provide a new set of reference points, continuing the algorithm according to the plan.

4.4.3 Avoiding known static obstacle

In the case of a completely static environment, the RRT algorithm only needs to determine the path once. All points along this path are suitable since there are no dynamic obstacles (such as human arms). The reference paths remain unchanged and will be guided entirely by AAPF until the path is completed.

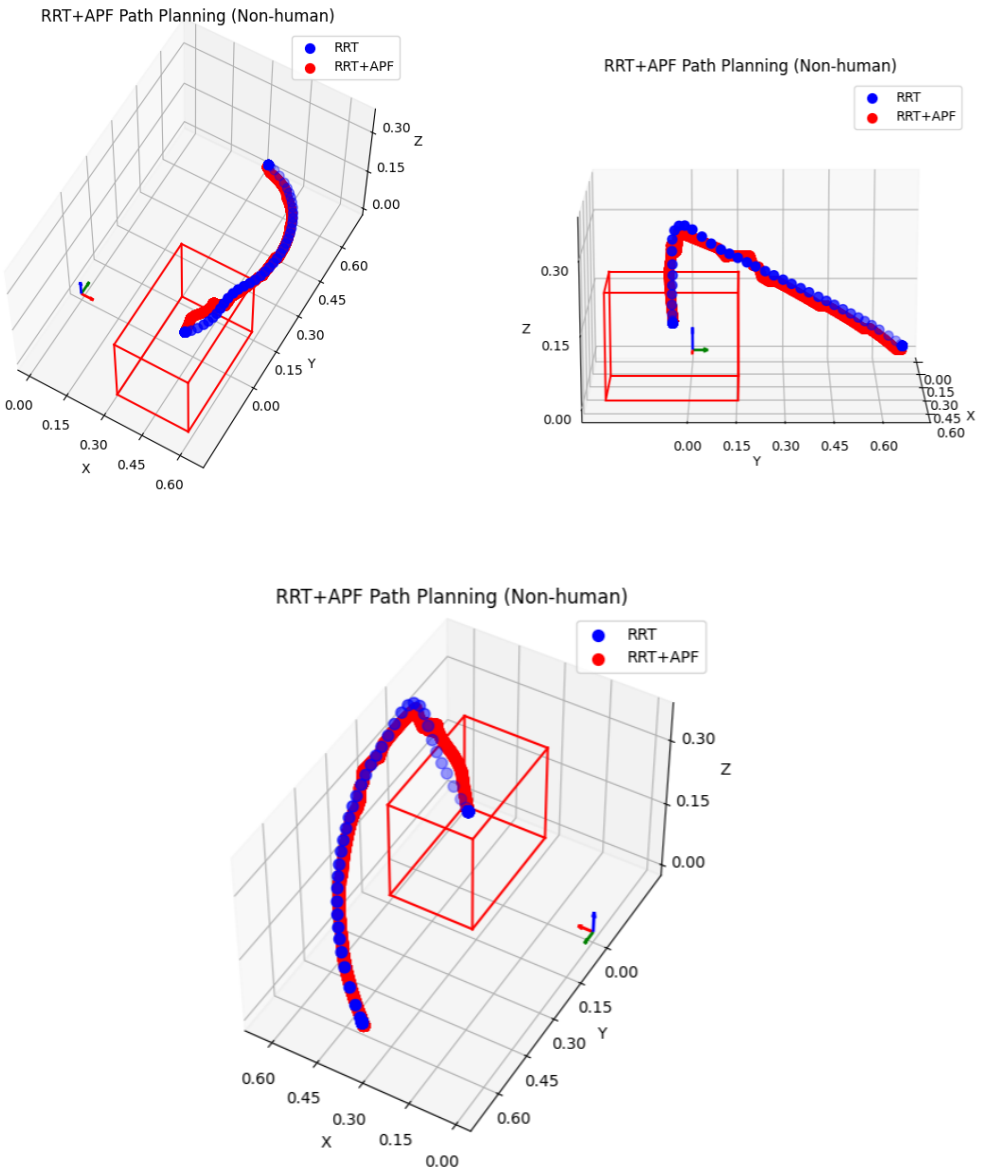

Figure 4.26: Path without dynamic obstacle

Where:

- *Blue path* is the set of point created from RRT - this is the reference path.
- *Red path* is the trajectory that the robot created and executed using the hybrid planner.

4.4.4 Avoiding human hands

Due to continuous information exchange between the Reference Manager and the AAPF planner, the environment is constantly updated. Upon receiving signals from the human wrist detection system, specifically the wrist position, the AAPF Planner generates a spherical obstacle with a radius of 21.8 cm and incorporates it into the PlanningScene. In this scenario, the human arm acts as an obstacle and is associated with a repulsive force field. The total forces will thus be adjusted, potentially pushing the path away from the obstacle to ensure safe navigation.

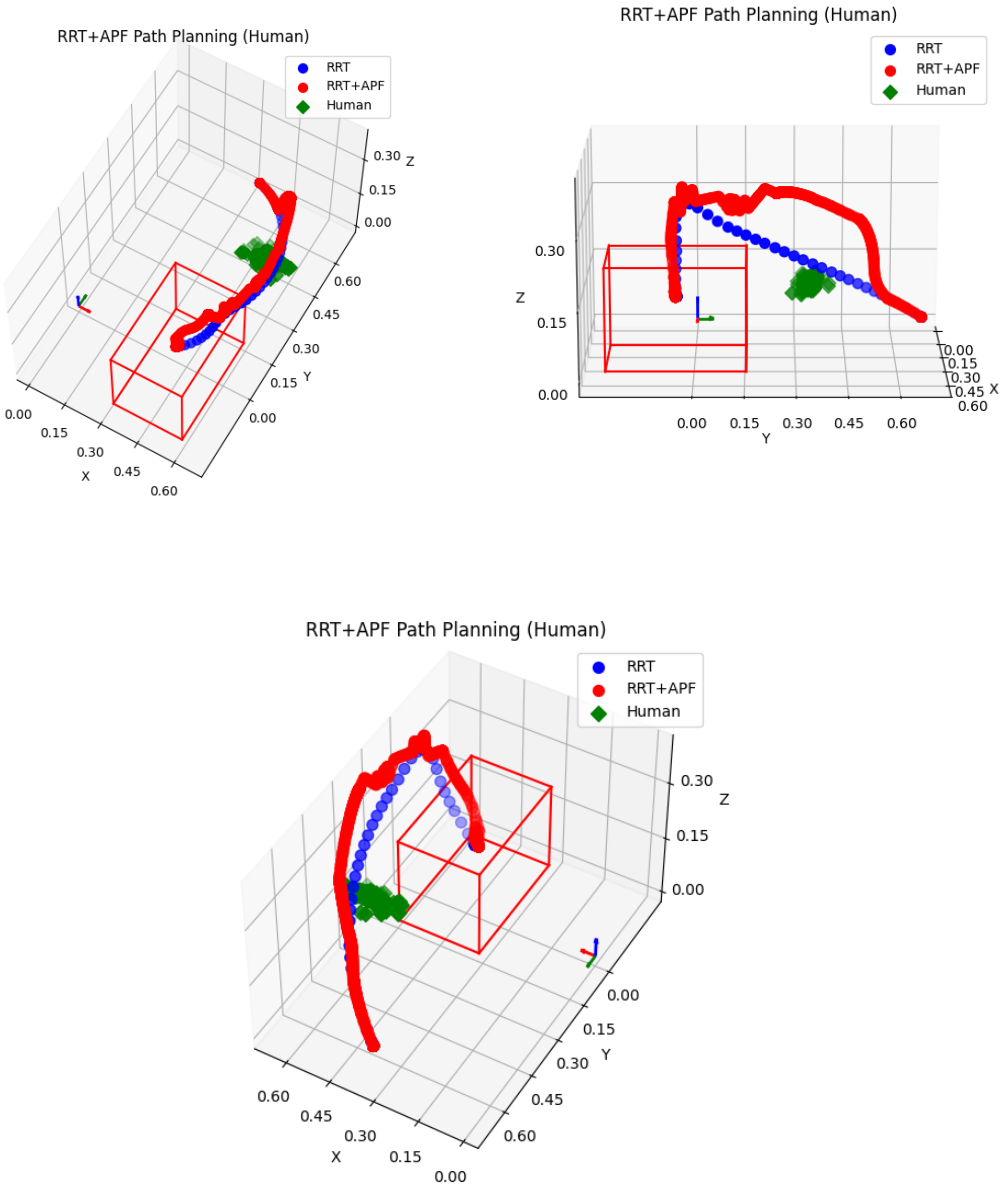

Figure 4.27: Path with human as obstacle

Trong đó:

- *Blue path* lis the set of point created from RRT - this is the reference path.
- *Red path* is the trajectory that the robot created and executed using the hybrid planner.
- *Green area* 3D position of the human wrist and hand.

4.5 Evaluation

In the case of static environment as in [4.26], the system can:

- Follow the reference path generated from the global planner RRT.
- The jerkiness of the path has reduced significantly.

In the case of dynamic environment as in [4.27], the system can:

- Identify dynamic obstacle (human wrist).

- Avoid the local minima.
- Update the trajectory when the reference path is blocked.
- Execute a different trajectory comparing to RRT.

However, the results also exhibit limitations: the robot moves slower compared to the RRT algorithm because it needs to stop at each waypoint along the reference path; furthermore, the path may exhibit jerky motion as the robot traverses regions influenced by multiple repulsive forces.

4.5.1 Parameters tuning

The team decided to tune the planner by leaving the gradient steps and thresholds unchanged. By changing k_{att} and k_{rep}, we came up with three sets of different parameters, yielding these results:

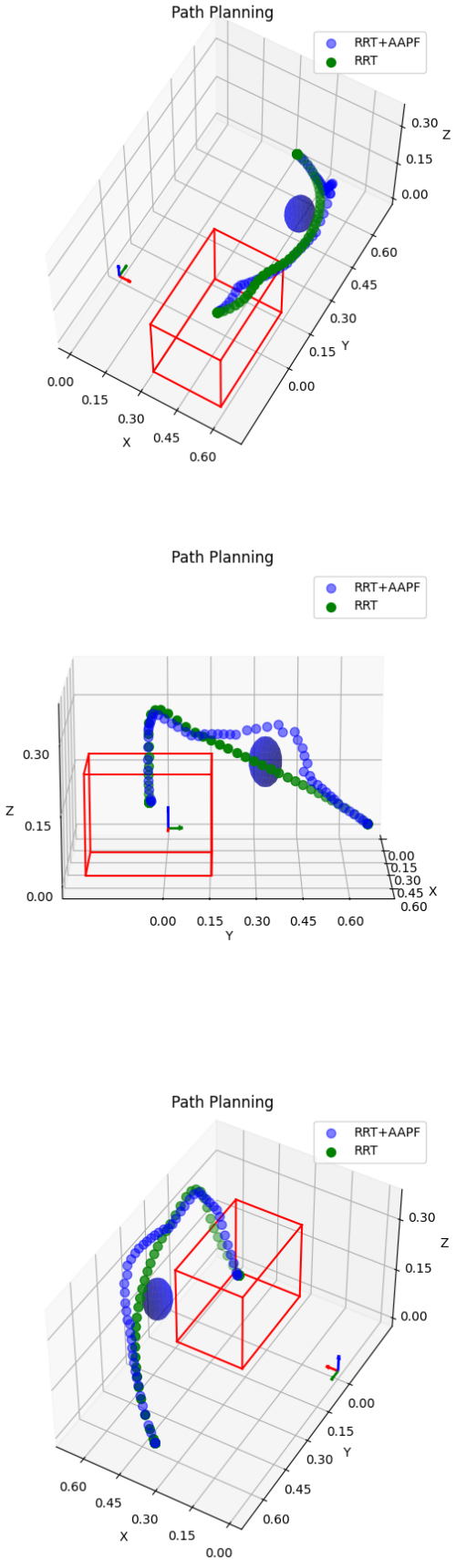

Figure 4.28: Path planned with the parameter pair 50-1

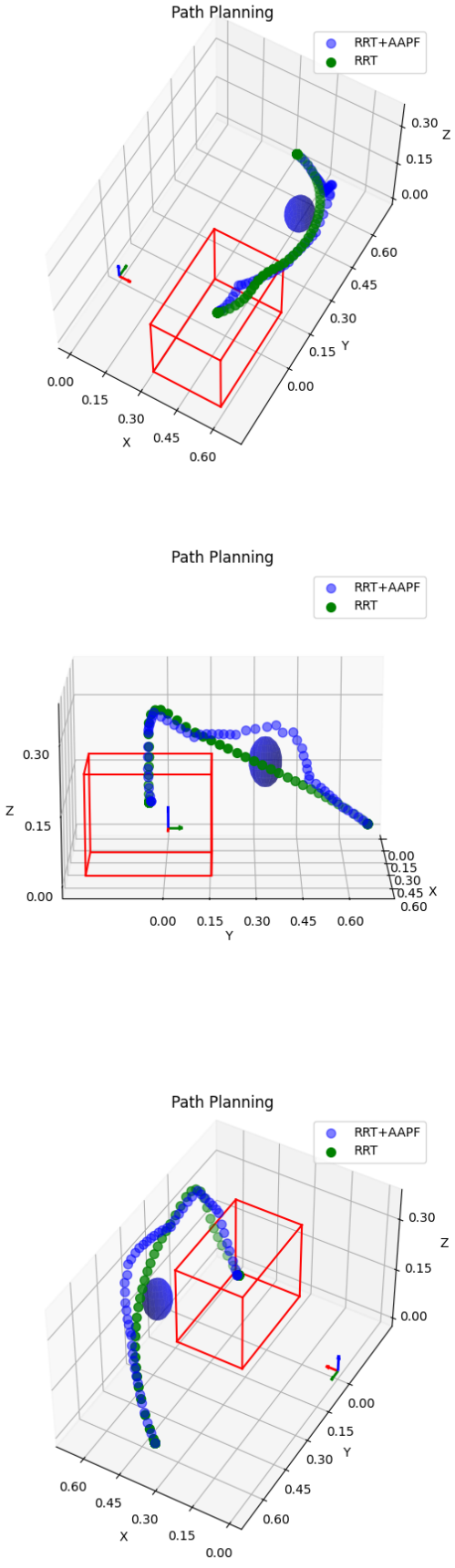

Figure 4.29: Path planned with the parameter pair 100-4

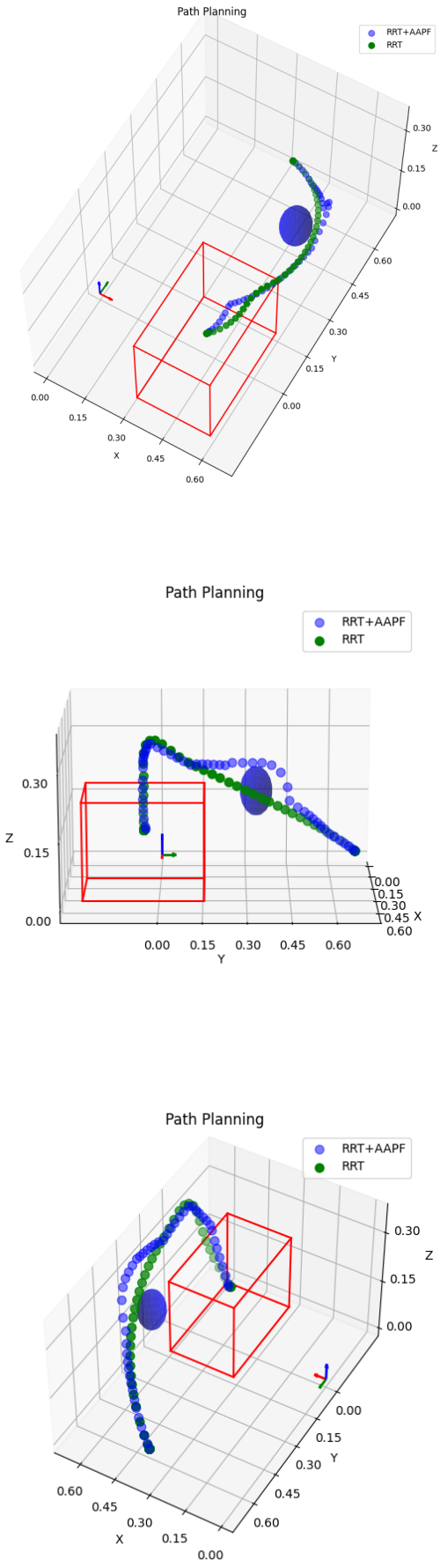

Figure 4.30: Path planned with the parameter pair 100-5

4.5.2 Joint Position graphs

After testing and experimenting, we recorded the data of the joint's trajectory. As the path was planned, the path was sent to the controller to move the robot. The value of each joints to time are:

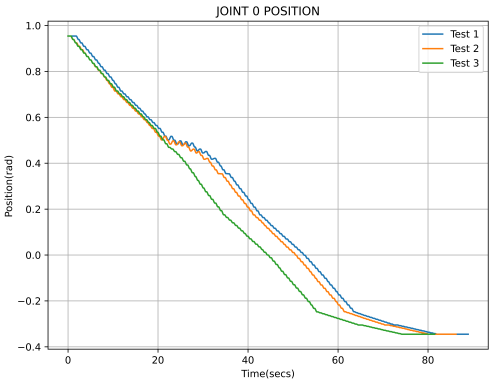

Figure 4.31: Joint 1 position

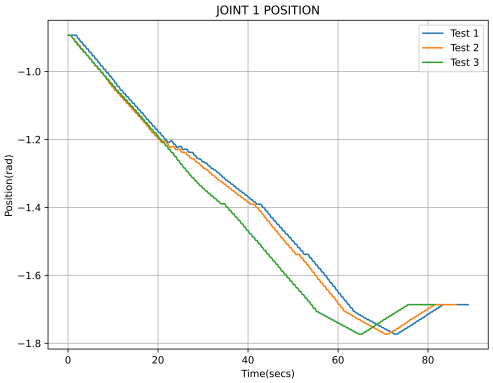

Figure 4.32: Joint 2 position

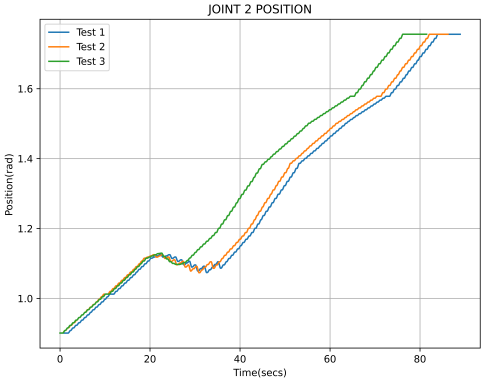

Figure 4.33: Joint 3 position

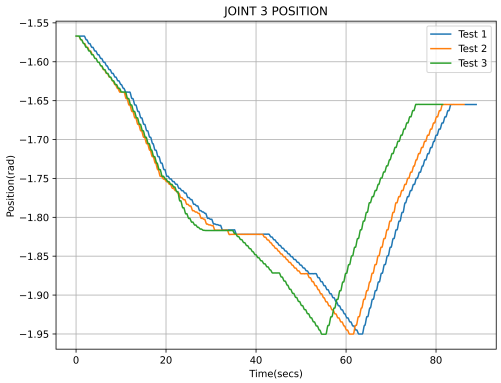

Figure 4.34: Joint 4 position

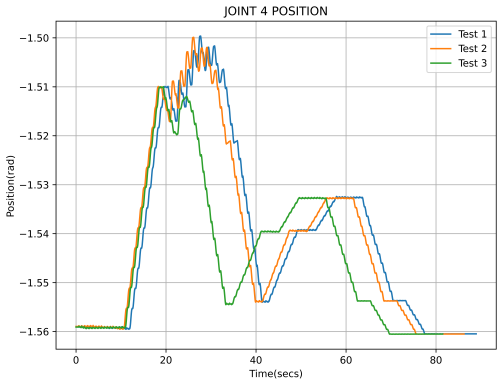

Figure 4.35: Joint 5 position

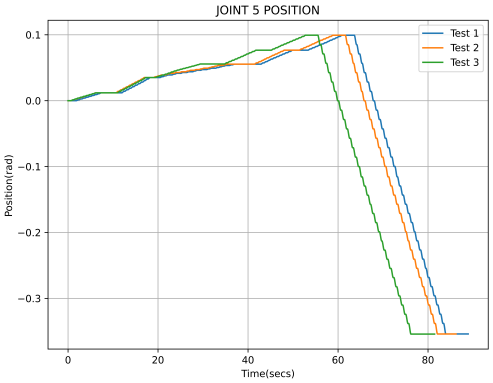

Figure 4.36: Joint 6 position

Where:

- Test 1: Testing with the parameter pair 1-1.
- Test 2: Testing with the parameter pair 200 - 400.
- Test 3: Testing with the parameter pair 500 - 150.

We can see that with the pair (500, 150), the planned path is returned without many jerkiness.

CHAPTER 5: Conclusion

5.1 Evaluation

The thesis has successfully completed the set goals, which are:

- Successfully built and designed a visual image processing system, allowing human recognition in space, extracting human location parameters through the application of artificial intelligence.
- Successfully built and designed a new planning set combining two algorithms RRT and AAPF. The new planning set achieved the expected results, while improving the disadvantages and limitations of the planning sets previously mentioned in the group's experiment.
- Ensuring the safety of humans and robots when working together in an environment, the robot has the ability to create and execute new trajectories to ensure the destination and avoid human hands.

Despite the above successes, the planner proposed by the group still encounters the following limitations:

- The time to execute the trajectory proposed by the group planner is still not fast, because the AAPF must stop to calculate the force each time a new destination is updated.
- The jerk of the trajectory is created by the new planning set, due to the inappropriate set of parameters, causing the robot to fall into areas suddenly affected by attractive and repulsive forces. This jerk can also be explained by the way the joint trajectory sent to the controller, so the velocity will not be a smooth line.
- The resolution and accuracy of the 3D Camera is also a factor that affects the accurate determination of the position of a person's hand in space.

In the course of conducting this project, our team acquired extensive knowledge and gained new experiences. Members engaged with the ROS robot operating system, utilized motion planning libraries such as MoveIt, processed data and simulated parameters in Matlab, interfaced with and controlled the UR5 robot, and explored modern robotics and control theories. This project provided an opportunity to review and apply the knowledge acquired at Hanoi University of Science and Technology. We extend our sincere gratitude to our mentors, Assoc. Prof. Ly Hoang Hiep, Prof. Bui Hai Le, and Prof. Mac Thi Thoa, for their dedicated guidance and support throughout this research endeavor. We also appreciate the Department of Electronics and the Faculty of Mechanical Engineering for providing the laboratory facilities, equipment, and robotic resources that enabled our team to conduct computations and experimental validations effectively on real systems.

5.2 Future development

To continue developing the interaction capabilities between humans and robots, in the future, our team aims to focus on research areas that aim to enhance motion planning, including:

- New control application: Replace the 'JointTrajectory' control method with 'JointVelocity' control to improve trajectory smoothness of the robot.
- Optimization of motion planning: Propose optimization methods for trajectory planning, such as time optimization, improving responsiveness, or trajectory tracking capabilities.

References

[1] Osher Azulay, Osher Azulay, Amir Shapiro, Amir Shapiro, Amir Shapiro, and Amir Shapiro. Wheel loader scooping controller using deep reinforcement learning. *IEEE Access*, 2021. doi: 10.1109/access.2021.3056625.

[2] Caio Cristiano Barros Viturino, Ubiratan de Melo Pinto Junior, André Gustavo Scolari Conceição, and Leizer Schnitman. Adaptive artificial potential fields with orientation control applied to robotic manipulators. *IFAC-PapersOnLine*, 53(2):9924–9929, 2020.

[3] Marcel Bogers, Marcel Bogers, Willem Horst, and Willem Horst. Collaborative prototyping: Cross-fertilization of knowledge in prototype-driven problem solving. *Journal of Product Innovation Management*, 2014. doi: 10.1111/jpim.12121.

[4] Cletis R. Booher and Betty S. Goldsberry. NASA's Man-Systems Integration Standards: A Human Factors Engineering Standard for Everyone in the Nineties. In *Dual-Use Space Technology Transfer Conference and Exhibition*, volume 1, Houston, TX, USA, May 1994. National Aeronautics and Space Administration. NASA Document ID: 19960021764.

[5] Thomas Böck and Thomas Bock. The future of construction automation: Technological disruption and the upcoming ubiquity of robotics. *Automation in Construction*, 2015. doi: 10.1016/j.autcon.2015.07.022.

[6] Chun-Hung Chen, Chun-Hung Chen, Jie Lin, Jianwu Lin, Enver Yücesan, Enver Yücesan, Stephen E. Chick, and Stephen E. Chick. Simulation budget allocation for further enhancing theefficiency of ordinal optimization. *Discrete Event Dynamic Systems*, 2000. doi: 10.1023/a:1008349927281.

[7] S.S. Ge and Y.J. Cui. New potential functions for mobile robot path planning. *IEEE Transactions on Robotics and Automation*, 16(5):615–620, 2000. doi: 10.1109/70.880813.

[8] Somayé Ghandi, Somayé Ghandi, Ellips Masehian, and Ellips Masehian. Review and taxonomies of assembly and disassembly path planning problems and approaches. *Computer-aided Design*, 2015. doi: 10.1016/j.cad.2015.05.001.

[9] Ross Girshick and Ross Girshick. Fast r-cnn. *null*, 2015. doi: 10.1109/iccv.2015.169.

[10] Elena Corina Grigore, Alessandro Roncone, Olivier Mangin, and Brian Scassellati. Preference-based assistance prediction for human-robot collaboration tasks. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4441–4448, 2018. doi: 10.1109/IROS.2018.8593716.

[11] Dominic Holzer and Dominic Holzer. Are you talking to me? why bim alone is not the answer. *null*, 2007. doi: null.

[12] Oussama Khatib. The potential field approach and operational space formulation in robot control. In *Adaptive and Learning Systems*, pages 367–377. Springer US, Boston, MA, 1986.

[13] Kay Kitazawa, Kay Kitazawa, Taku Fujiyama, and Taku Fujiyama. Pedestrian vision and collision avoidance behavior: Investigation of the information process space of pedestrians using an eye tracker. *null*, 2010. doi: 10.1007/978-3-642-04504-2_7.

[14] Aadi Kothari, Tony Tohme, Xiaotong Zhang, and Kamal Youcef-Toumi. Enhanced human-robot collaboration using constrained probabilistic human-motion prediction, 2023. URL https://arxiv.org/abs/2310.03314.

[15] Daniel Kruse, Daniel Kruse, Richard J. Radke, Richard J. Radke, John T. Wen, and John T. Wen. Collaborative human-robot manipulation of highly deformable materials. *null*, 2015. doi: 10.1109/icra.2015.7139725.

[16] Meng-Lun Lee, Meng-Lun Lee, Sara Behdad, Sara Behdad, Xiao Liang, Xiao Liang, Minghui Zheng, and Minghui Zheng. Task allocation and planning for product disassembly with human–robot collaboration. *Robotics and Computer-integrated Manufacturing*, 2022. doi: 10.1016/j.rcim.2021.102306.

[17] Ci-Jyun Liang, Ci-Jyun Liang, Xi Wang, Xi Wang, Xi Wang, Vineet R. Kamat, Vineet R. Kamat, Carol C. Menassa, and Carol C. Menassa. Human–robot collaboration in construction: Classification and research trends. *Journal of Construction Engineering and Management-asce*, 2021. doi: 10.1061/(asce)co.1943-7862.0002154.

[18] Tsung-Yi Lin, Tsung-Yi Lin, Michael Maire, Michael Maire, Serge Belongie, Serge Belongie, James Hays, James Hays, Pietro Perona, Pietro Perona, Deva Ramanan, Deva Ramanan, Piotr Dollár, Piotr Dollar, C. Lawrence Zitnick, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. *null*, 2014. doi: 10.1007/978-3-319-10602-1_48.

[19] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[20] Kelly Merckaert, Bryan Convens, Marco M Nicotra, and Bram Vanderborght. Real-time constraint-based planning and control of robotic manipulators for safe human–robot collaboration. *Robot. Comput. Integr. Manuf.*, 87(102711):102711, June 2024.

[21] Thibaut Munzer, Marc Toussaint, and Manuel Lopes. Efficient behavior learning in human–robot collaboration. *Auton. Robots*, 42(5):1103–1115, June 2018.

[22] Luka Peternel, Luka Peternel, Nikos G. Tsagarakis, Nikos G. Tsagarakis, Darwin G. Caldwell, Darwin G. Caldwell, Arash Ajoudani, and Arash Ajoudani. Robot adaptation to human physical fatigue in human—robot co-manipulation. *Autonomous Robots*, 2018. doi: 10.1007/s10514-017-9678-1.

[23] Kamel S. Saidi, Kamel S. Saidi, Thomas Böck, Thomas Bock, Christos Georgoulas, and Christos Georgoulas. Robotics in construction. *null*, 2016. doi: 10.1007/978-3-319-32552-1_57.

[24] Hang Su, Hang Su, Subhransu Maji, Subhransu Maji, Evangelos Kalogerakis, Evangelos Kalogerakis, Erik Learned-Miller, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. *null*, 2015. doi: 10.1109/iccv.2015.114.

[25] Matthew R. Walter, Matthew R. Walter, Matthew Antone, Matthew Antone, Matthew Antone, Ekapol Chuangsuwanich, Ekapol Chuangsuwanich, Ekapol Chuangsuwanich, Andrew Correa, Andrew Correa, Randall Davis, Randall Davis, Luke Fletcher, Luke Fletcher, Emilio Frazzoli, Emilio Frazzoli, Yuli Friedman, Yuli Friedman, James Glass, James Glass, Jonathan P. How, Jonathan P. How, Jeong hwan Jeon, Jeong hwan Jeon, Sertaç Karaman, Sertac Karaman, Brandon D. Luders, Brandon D. Luders, Nicholas Roy, Nicholas Roy, Stefanie Tellex, Stefanie Tellex, Seth

Teller, and Seth Teller. A situationally aware voice-commandable robotic forklift working alongside people in unstructured outdoor environments. *Journal of Field Robotics*, 2015. doi: 10.1002/rob.21539.

[26] Haitao Wu, Haitao Wu, Heng Li, Heng Li, Yantao Yu, Xin Fang, Xin Fang, Xiaochun Luo, and Xiaochun Luo. A survey on teaching workplace skills to construction robots. *Expert Systems With Applications*, 2022. doi: 10.1016/j.eswa.2022.117658.

[27] Bo Xiao, Bo Xiao, Chen Chen, Chen Chen, Xianfei Yin, and Xianfei Yin. Recent advancements of robotics in construction. *Automation in Construction*, 2022. doi: 10.1016/j.autcon.2022.104591.

[28] Sangseok You, Sangseok You, Jeonghwan Kim, Jeonghwan Kim, Sang Hyun Lee, SangHyun Lee, Vineet R. Kamat, Vineet R. Kamat, Lionel Robert, Lionel + "Jr" Robert, and Lionel P. Robert. Enhancing perceived safety in human–robot collaborative construction using immersive virtual environments. *Automation in Construction*, 2018. doi: 10.1016/j.autcon.2018.09.008.

[29] Xingwei Zhao, Yiming Chen, Lu Qian, Bo Tao, and Han Ding. Human–robot collaboration framework based on impedance control in robotic assembly. *Engineering (Beijing)*, 30:83–92, November 2023.