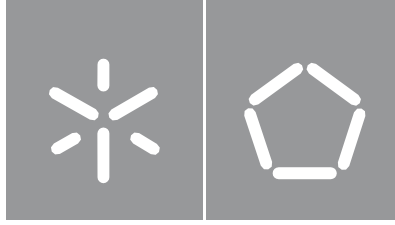




Universidade do Minho
Escola de Engenharia

António João Gonçalves Matos

**Motion planning for robotic arms integrated on
mobile platforms for internal logistics tasks**



Universidade do Minho
Escola de Engenharia

António João Gonçalves Matos

**Motion planning for robotic arms integrated on
mobile platforms for internal logistics tasks**

Master's Dissertation
Master's in Mechanical Engineering
Specialization in Mechatronic Systems

Work developed under the supervision of:
Professor José Mendes Machado
Professor Luís Filipe Castro Freitas Louro

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada. Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho



Atribuição

CC BY

<https://creativecommons.org/licenses/by/4.0/>

Acknowledgments

Firstly, I would like to express my deepest gratitude to my supervisor José Machado, for his support throughout the development of this work and to Professor Luís Louro for his invaluable guidance in all matters relating to robotics.

I want to express my heartfelt thanks to Professor Estela Bicho for her generosity in sharing knowledge and for her willingness to help me whenever I needed it.

A special acknowledgment goes to the team involved in this project, namely Professor Sérgio Monteiro and my colleagues Ivan Castro, João Veríssimo, João Diogo, and João Araújo, with special attention to the latter for putting me through this delicate phase.

I am also thankful to the Bosch team for their collaboration with the University of Minho, with special recognition to Tiago Carvalho and Damásio Eurico for their dedication during our regular meetings to monitor the project's progress.

I am grateful to the entire MarLab team at the Industrial Electronics Department for warmly welcoming me during these past months and for providing the tools and resources needed to accomplish this work.

I also want to express my gratitude to Paulo Bourbon from Europneumaq for allowing me to conduct tests with the gripper used in this project.

I appreciate my family, whose unwavering support has been my rock during this challenging period. Their encouragement was fundamental.

Thank you to the Department of Mechanical Engineering professors for the knowledge imparted during my academic journey over the past five years.

Lastly, I want to thank my friends who shared this challenging journey with me. Special thanks to the Roubocopo group.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

Abstract

The developed work in this dissertation is part of the MIAR project, which focuses on using synchronized Mobile Intelligent Autonomous Robots (MIAR) to automate logistics operations at Bosch Car Multimedia BrgP. The autonomous mobile manipulator must perform pick-and-place operations involving *Targets*, boxes containing certain electronic products, between a storage buffer and access ramps connected to the assembly lines. The purpose of this dissertation was to develop motion planning for the KUKA LBR iiwa 14 R820 manipulator to execute the intended tasks. The use of a redundant arm in this project aimed to create collision-free trajectories in confined spaces. Firstly, a simulation model was created at CoppeliaSim to represent the Bosch shop floor where the mobile manipulator will operate. Under this scenario, several tasks were performed, such as determining the number of boxes the manipulator could handle from the dollies and checking the maximum number of boxes that could be transported on top of the mobile platform. To carry out the planning tests, the collaborative gripper 2FGP20 was selected to meet the project's requirements, and custom-designed fingers were simulated at Autodesk Fusion to withstand the imposed loads during manipulation. Additionally, new optimized ramps were introduced to ensure that the arm could access boxes.

A method of communication between CoppeliaSim and MoveIt, software used for trajectory generation, was implemented. This communication was established using ROS2 via two launch files. A benchmarking study was conducted among ten planners, identifying T-RRT as the most suitable for the project. Subsequently, the planner was customized to ensure better results across diverse manipulation scenarios, and a post-processor was utilized to verify if it would lead to significant improvements, considering the project's stringent requirements.

In conclusion, the manipulator demonstrated satisfactory performance, given the complexity of its working environment. Some limitations were identified, and suggestions for future improvements were proposed.

Keywords: Global Planners, Motion Planning, Pick-and-Place, Smoothness.

Resumo

O trabalho desenvolvido nesta dissertação enquadra-se no projeto MIAR, que se foca na utilização de Robôs Autónomos Inteligentes Móveis (MIAR) sincronizados para automatizar as operações de logística na Bosch Car Multimedia BrgP. O manipulador móvel autónomo necessita de realizar operações de *pick-and-place* envolvendo *Targets*, caixas contendo determinados produtos eletrónicos, entre um *buffer* de armazenamento e rampas de acesso ligadas às linhas de montagem. O objetivo desta dissertação foi desenvolver planeamentos de movimento para o manipulador KUKA LBR iiwa 14 R820 poder executar as tarefas pretendidas. A utilização de um braço redundante neste projeto foi com a intenção de se criar trajetórias livres de colisões em espaços confinados. Primeiramente, foi criado um modelo de simulação no CoppeliaSim para representar o chão de fábrica na Bosch, no qual o manipulador móvel irá operar. Neste cenário, foram realizadas diversas tarefas, tais como determinar a quantidade de caixas que o manipulador poderia alcançar dos *dollies* e verificar a quantidade máxima de caixas que poderiam ser transportadas em cima da plataforma móvel. Para a realização dos testes de planeamento, foi selecionado o *gripper* colaborativo 2FGP20, de forma a satisfazer os requisitos do projeto, e foram simulados dentes personalizados no Autodesk Fusion para suportar as cargas impostas durante a manipulação. Além disso, foram introduzidas novas rampas otimizadas para garantir que as caixas pudessem ser acedidas pelo braço.

Foi desenvolvido um método de comunicação entre o CoppeliaSim e o MoveIt, software que é utilizado para a geração de trajetórias. Esta comunicação foi estabelecida por ROS2 através de dois *launch* files. Um estudo de benchmarking entre dez planeadores foi realizado, identificando o T-RRT como o mais adequado para o projeto. Posteriormente, o planeador foi personalizado de forma a garantir melhores resultados em diversos cenários de manipulação, e um pós-processador foi adicionado para verificar a existência de melhorias significativas, perante os rigorosos requisitos do projeto.

Concluiu-se que o manipulador demonstrou um desempenho satisfatório perante a complexidade do ambiente de trabalho do projeto. Foram ainda identificadas algumas limitações e propostas sugestões de melhorias futuras.

Palavras-chave: *Pick-and-Place*, Planeadores Globais, Planeamento de Movimentos, Suavidade.

Contents

Abstract	iv
Resumo	v
1 Introduction.....	1
1.1 Motivation	2
1.2 Objectives	3
1.3 Structure of the Dissertation	4
2 State of the Art	6
2.1 The Importance of Collaborative Robots in Industry	6
2.2 Automatic Systems for Feeding and Collecting Boxes from Production Lines	9
2.3 Motion Planners	14
2.3.1 Global Planners	15
2.3.2 Local Planners.....	24
2.4 Motion Planning and Control for Autonomous Mobile Manipulators.....	28
3 Theoretical Foundations and Development Tools	31
3.1 Fundamental Concepts of Motion Planning.....	31
3.2 Motion Planners Utilized in the MIAR Project	33
3.2.1 PRM*	34
3.2.2 LazyPRM*	35
3.2.3 KPIECE.....	35
3.2.4 T-RRT	36
3.2.5 Bi-TRRT	37
3.2.6 BiEST	38
3.2.7 ProjEST.....	39
3.3 Hardware Components	39

3.3.1	Forward Kinematics.....	40
3.3.2	Inverse Kinematics	49
3.4	Software Tools.....	53
3.4.1	ROS2 Middleware.....	53
3.4.2	Movelt Library	59
3.4.3	CoppeliaSim.....	64
3.4.4	Autodesk Fusion	66
4	Developed Work.....	67
4.1	System Architecture	67
4.2	Simulation Model	68
4.3	Selection of the Gripper	73
4.4	Development of the Gripper Fingers.....	78
4.4.1	Modeling the Gripper Fingers	79
4.4.2	Evaluation and simulation of the fingers	80
4.4.3	Fingers Test	84
4.5	Maximum Number of <i>Targets</i> to be Manipulated	86
4.6	Configuration in Movelt	90
4.6.1	Launch KUKA LBR iiwa 14 R820 + 2FGP20 in Rviz	92
4.6.2	Movelt-CoppeliaSim Communication	94
4.7	Ramp Analysis and Optimization.....	97
4.8	Pick-and-Place Operations in Movelt/CoppeliaSim	102
5	Analysis of Results.....	109
5.1	Selection of the Ideal Motion Planner.....	109
5.2	Analysis of the Motion Planners Comparison	112
5.3	Optimization of the Motion Planner	116
5.4	Manipulation Operations Using the Selected Motion Planner	118

6	Conclusions.....	121
6.1	Future work	123
	References.....	125
	Annex.....	i
A.	Workspace Graphic for the KUKA LBR iiwa 14 R820.....	i
	Appendix.....	ii
A.	Technical Drawings of the “As Is” Access Ramps.....	ii
B.	Technical Drawings of the Developed Fingers	vi
C.	Nodes and Topics Responsible for the Transmission of the Manipulator’s Joint States Between MoveIt and CoppeliaSim.....	viii
D.	Nodes and Topics Responsible for the Transmission of the Objects’ Poses Between MoveIt and CoppeliaSim	viii
E.	Technical Drawings of the “To Be” Access Ramps	ix
F.	Benchmarking Study of the 10 OMPL planners	xi
G.	Performance Comparison Between Custom T-RRT Planners.....	xvi

List of Figures

Figure 1 - (a) Annual installations of Industrial robots - World; (b) Annual Installations of Industrial robots by customer industry – World. Adapted from (IFR International Federation of Robotics, 2023).	6
Figure 2 - Installations per year of collaborative and traditional industrial robots, 2017 to 2022. Adapted from (IFR International Federation of Robotics, 2023).	8
Figure 3 - Example of an AS/RS layout. Adapted from (Xing et al., 2010).	10
Figure 4 - Example of an AutoStore system. Adapted from (Trost et al., 2022).	11
Figure 5 - Example of a warehouse scenario where a mobile manipulator performs pick and place tasks. Adapted from (Thakar et al., 2022).	12
Figure 6 - Autonomous Case-handling Robotic System (ACR) from HAI ROBOTICS. Adapted from (H. Robotics, 2022).	13
Figure 7 - Skyler and Cart robots from Caja Robotics. Adapted from (C. Robotics, 2023).	13
Figure 8 - Example of building a roadmap on a robot. Obstacles are represented in blue. Adapted from (Eric O. Scott, 2015).	16
Figure 9 - Example of the building of an RRT tree. Adapted from (Barland, 2012).	17
Figure 10 - (a) Kinematic structure of a robot and human arm and their respective key positions; (b) Correspondence method wrist-elbow-in-line. Adapted from (W. Liu et al., 2017).	22
Figure 11 - (a) Direct movement from start to target; (b) Composite movement; (c) Division of the different phases of the pick and place movement. Adapted from (Gulletta et al., 2021).	24
Figure 12 - Example of a path taken by a robot using the potential field method. Adapted from (Aslan, 2021).	25
Figure 13 - Block scheme of the Hybrid Joint/Cartesian DMP-Based Approach. Adapted from (Lauretti et al., 2019).	27
Figure 14 – Schematic representation of robot motion planning. Adapted from (S. Liu & Liu, 2020).	32
Figure 15 - The representation of a robot's tool frame in relation to the base frame. Adapted from (Craig, 2017).	33

Figure 16 - Example of a construction process of the Transition-based RRT planner. The red zones represent high cost area of the cost map. Adapted from (Jaillet et al., 2010).	37
Figure 17 - Joint angles of the KUKA LBR iiwa 14 R820.	40
Figure 18 - Assignment of the D-H coordinate frames to each joint of the KUKA LBR iiwa 14 R820 robot.	41
Figure 19 - Representation of the arm angle parameter in the redundancy circle. Adapted from (Alberto & Borges, 2022).	45
Figure 20 - Illustration of the virtual manipulator used to calculate FK. Adapted from (Faria et al., 2018).	47
Figure 21 - Languages used in ROS communication tools. Adapted from (The Robotics Back-End, 2018).	54
Figure 22 - Workspace with CMake and Python packages.	55
Figure 23 - Illustrative example of communication between nodes grouped into different packages. Adapted from (The Robotics Back-End, 2018).	56
Figure 24 - Example of a srv file.	57
Figure 25 - Communication between nodes by a topic and a service. Adapted from (Open Robotics, 2024b).	57
Figure 26 - Communication between nodes by an action. Adapted from (Open Robotics, 2024a).	58
Figure 27 - Initial window of the MoveIt Setup Assistant. Adapted from (PickNik Robotics, 2024c).	61
Figure 28 - Example of a config folder from the moveit_config package generated by the MoveIt Setup Assistant.	61
Figure 29 - High-level system architecture of the MoveIt move_group node. Adapted from (PickNik Robotics, 2024a).	63
Figure 30 - CoppeliaSim interface.	66
Figure 31 - Autodesk Fusion Interface.	66
Figure 32 - Global architecture of the MIAR system.	67
Figure 33 - Intended work layout at Bosch Car Multimedia BrgP.	69
Figure 34 – Containers-RAKO from Utz Group and their respective CAD models.	70
Figure 35 - (a) Outer dimensions of the base of the larger container; (b) Outer dimensions of the base of the smaller container.	70

Figure 36 - Dolly Supply zone for MIAR Project.	71
Figure 37 – Dolly Channel and CAD representation.....	71
Figure 38 - One-level access ramp for the storage of Targets.	72
Figure 39 - Two-level access ramp for the storage of Targets.	72
Figure 40 - Representation of the work layout using the CoppeliaSim software.....	73
Figure 41 - Systematic Comparison between criteria.	77
Figure 42 - Attribution of weights relative to criteria.	77
Figure 43 - Weighted objectives evaluation chart for five gripper solutions.	77
Figure 44 - 2FGP20 front view (dimensions in mm). Adapted from (OnRobot, 2023).	78
Figure 45 - Profile view of both Targets with the outer dimensions.	79
Figure 46 – (a) Maximum Gripper Aperture; (b) Minimum Gripper Aperture.	80
Figure 47 - Gripper holding the two types of Targets.....	80
Figure 48 - Pre-processing of the mobile gripper finger.	81
Figure 49 - Mesh generation parameters in Autodesk Fusion.....	82
Figure 50 – (a) Static simulation on the movable finger; (b) Static simulation on the fixed finger.	83
Figure 51 - Fingers shape optimization.	84
Figure 52 - Static simulation of the boxes to be manipulated.	84
Figure 53 - (a) Grip test of the 400x300 box; (b) Grip test of the 600x400 box.....	85
Figure 54 - Payload diagram of KUKA LBR iiwa 14 R820 manipulator. Adapted from (KUKA, 2019).....	85
Figure 55 – (a) Manipulator grasping the first box from the dolly; (b) Maximum number of Targets placed on a dolly that the arm can reach.....	86
Figure 56 – Study of the maximum number of 600x400 boxes on the platform.	87
Figure 57 - Study of the maximum number of 400x300 boxes on the platform.	87
Figure 58 - Study of the maximum number of boxes of both types on the platform.	88
Figure 59 - Conceptual Structure to transport the boxes at the KMR.	88
Figure 60 - Minimum height required for the rear framework that supports the larger boxes.	89
Figure 61 – (a) Top view of the retention slot at the KMR base for securing small boxes; (b) Top view of the retention slot in the rear structure of the KMR for securing the stacked large boxes.	89

Figure 62 - (a) Maximum number of 400x300 boxes positioned at the platform's base; (b) Manipulator reaching the fourth 400x300 box.....	90
Figure 63 - Study of the maximum number of 600x400 boxes at the rear of the platform's structure.....	90
Figure 64 - ROS2 Gripper package generated from Autodesk Fusion.	91
Figure 65 - Some configuration steps for the robotic arm in MoveIt Setup Assistant.	92
Figure 66 - KUKA LBR iiwa 14 R820 with 2FGP20 gripper in Rviz.....	94
Figure 67 – MoveIt-CoppeliaSim Communication Diagram.....	94
Figure 68 - CoppeliaSim-MoveIt Communication.	97
Figure 69 - Attempt to collect or place Targets by the manipulator on the different levels of the ramp.....	98
Figure 70 - (a) Collision of the manipulator's links with the ramp upper level; (b) Proposed staircase ramp.	99
Figure 71 - "To be" one-level ramp.	99
Figure 72 - Picking the 600x400 box on the optimized ramp.	100
Figure 73 - (a) Collision between the robotic arm and the two-level ramp with entrance cuts; (b) Robotic arm retrieving an empty box from the two-level ramp with entrance cuts.	101
Figure 74 - Box placement and retrieval across different levels of the optimized two-level ramp.	101
Figure 75 - Example of an optimized two-level ramp.	102
Figure 76 - Diagram of the procedure for initializing the Pick-and-Place Operations.....	104
Figure 77 - General flowchart of the Pick-and-Place operation.	105
Figure 78 - Flowchart of the Pick operation.....	107
Figure 79 - Flowchart of the Place operation.....	108
Figure 80 - Comparison of planning and execution times for the place operation of the 600x400 boxes.	112
Figure 81 - Comparison of the path length for the place operation of the 600x400 boxes..	113
Figure 82 - Comparison of the smoothness factor for the place operation of the 600x400 boxes.	114
Figure 83 - Number of attempts for the place operation of the 600x400 boxes.	114

Figure 84 - (a) Picking 600x400 boxes from the ramp. The task can be seen in: https://youtu.be/4_dr845m8Tw ; (b) Picking 400x300 boxes from the ramp. The task can be seen in: https://youtu.be/8CKV2Hu8QAo	119
Figure 85 - (a) Placing 600x400 boxes in the ramp. The task can be seen in: https://youtu.be/a5ITRRm8M_8 ; (b) Placing 400x300 boxes in the ramp. The task can be seen in: https://youtu.be/GSF92bFSylc	119
Figure 86 - (a) Picking the <i>Targets</i> from the Dolly Supply. The picking task for the large boxes can be seen in: https://youtu.be/0hr2zuzmehs . The picking task for the small boxes can be seen in: https://youtu.be/aKFyQ0qsOxU ; (b) Placing the <i>Targets</i> in the Dolly Supply. The placing task for the large boxes can be seen in: https://youtu.be/kp6llsDKCKc . The placing task for the small boxes can be seen in: https://youtu.be/3-6wfUqV1q8	120
Figure 87 - Number of attempts for the pick operation of the 600x400 boxes.	xii
Figure 88 - Number of attempts for the place operation of the 400x300 boxes.	xiii
Figure 89 - Number of attempts for the pick operation of the 400x300 boxes.	xv

List of Tables

Table 1 - Motion range and speed specifications of the KUKA LBR iiwa 14 R820 joints.	40
Table 2 - Denavit-Hartenberg parameters table for the KUKA LBR iiwa 14 R820 robot.	42
Table 3 - Functional and non-functional requirements for MIAR project gripper.	75
Table 4 - Comparison of product attributes.....	76
Table 5 - Topics and messages used for the MoveIt-CoppeliaSim communication.	96

List of Acronyms

Acronym	Description
AMR	Autonomous Mobile Robot
AS/RS	Automated Storage and Retrieval System
BrgP	Braga Plant
BP	Bounce Posture
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CHOMP	Covariant Hamiltonian Optimization for Motion Planning
D-H	Denavit-Hartenberg
DMP	Dynamic Movement Primitive
DoF	Degrees of Freedom
EST	Expansive Space Trees
FK	Forward Kinematics
GC	Global Configuration
GTP	Good-to-Person
HRC	Human-Robot Collaboration
HUMP	Human-like Upper-limb Motion Planner
IK	Inverse Kinematics
KMR	Kuka Mobile Robot
KPIECE	Kinodynamic Planning by Interior-Exterior Cell Exploration
LiDAR	Light Detection and Ranging
MIAR	Mobile Intelligent Autonomous Robots
OMPL	Open Motion Planning Library
PRM	Probabilistic Roadmap Method
ROS	Robot Operating System
RRT	Rapidly Exploring Random Tree
RULA	Rapid Upper Limb Assessment
SEW	Shoulder-Elbow-Wrist
SEWv	Virtual Shoulder-Elbow-Wrist

SKU	Stock-Keeping Unit
SP	Start posture
SRDF	Semantic Robot Description Format
S-R-S	Spherical-Rotational-Spherical
STOMP	Stochastic Trajectory Optimization for Motion Planning
TAP	Target Arm Pose
TEB	Time Elastic Band
TP	Target Posture
T-RRT	Transition-based Rapidly Exploring Random Tree
URDF	Unified Robot Description Format
XML	Extensible Markup Language
YAML	Yet Another Markup Language

1 Introduction

At Bosch Car Multimedia Braga Plant (BrgP), as well as in other factories within the group, the movement of boxes containing electronic products, known as *Targets*, across the shop floor between various assembly lines is a complex task. Currently, *Targets* are moved manually, a time-consuming process that often results in missed delivery deadlines. This issue has several implications for operational performance and production costs.

The present dissertation is part of the “Connected Manufacturing - Digital Transformation” Program (ref: SIFN-01-9999-FN-179826 – I&D) of the Bosch & UMinho partnership, more specifically the Project CM.5G.P16 – MIAR – “Flexible system to transport products between shared processes with Mobile Intelligent Autonomous Robots (MIAR) synchronized with production lines”. The MIAR project aims to develop intelligent, flexible, and automated solutions that enable the transport and positioning of products within established deadlines, relying on a combination of advanced technologies and solutions in areas such as collaborative robotics, automation, computer vision, and IoT.

Within the scope of the MIAR project, it is necessary to address the previously mentioned situation by implementing innovative technologies. The goal is to move the *Targets* from manual to automated through solutions that guarantee faster procedures, respecting predefined timeframes. The solutions to be implemented aim to use synchronized autonomous mobile manipulator robots integrated with production lines to perform pick, transport, and place operations for the *Targets*. For this purpose, vision systems are required to identify the *Targets* and potential obstacles to the movement of the autonomous mobile robot, such as humans, since the environment in which the robot will operate is shared with factory workers. Furthermore, since the intention is to use more than one autonomous mobile manipulator robot in the established operations, it is important to ensure that no collisions occur between them during the execution of tasks. To achieve this, it is necessary to develop a flexible fleet management system designed for the respective robots. It is also essential to control the movement of the redundant anthropomorphic robotic arm mounted on the mobile platform to ensure the successful execution of pick, transport, and place tasks.

In the context of my dissertation, the goal is to develop and test methods for generating collision-free pick-and-place movements for the robotic arm integrated into the mobile robot. The robotic arm is responsible for picking the *Targets* from the storage buffer and placing them onto the access ramps leading to the assembly lines. It must also collect empty boxes from the ramps and return them to the storage buffer. As these robots must coexist with human operators, in addition to safety, an important requirement is that their movements are legible to make the robot's behavior more transparent to the workers around them.

1.1 Motivation

Due to the constant technological evolution in Industry 4.0, the use of automated systems to control industrial processes has grown. The main purpose is to maximize the productivity and flexibility of the processes by integrating autonomous machines in monotonous and hazardous operations and reducing human intervention (NECULA et al., 2022). Currently, tasks such as transporting and handling components across production lines remain challenging in manufacturing environments. The increasing demand for adaptable workspaces requiring more flexible tasks has increased the need for collaborative robots in the industry. The use of collaborative robots in shared environments with humans enables them to assist workers in various tasks, particularly repetitive ones (Weidemann et al., 2023).

The MIAR project is concerned with implementing a promising approach to ensure the precise and safe placement and collection of boxes in the production lines. This is achieved by integrating an omnidirectional mobile platform equipped with a redundant collaborative manipulator. This system is designed to work with human workers in a shared industrial space. As such, the robotic arm's movements must be safe and human-readable during all the pick-and-place operations.

Moreover, the robotic arm's environment is expected to be filled with access ramps that are presented as obstacles to the manipulator's movements. The robot must generate collision-free trajectories to precisely deliver the boxes to the designated ramps without colliding with them during the entire process. Consequently, the study of the redundancy of the manipulator with the mobile platform's position will bring good benefits to the execution of the intended operations. The proposed goal brings the opportunity to realize an in-depth

analysis of the available motion planning techniques to create a smooth and efficient trajectory throughout all operational phases.

1.2 Objectives

The main objective of this dissertation is to develop methods and algorithms for generating safe and precise movements for the robotic arm, ensuring the accurate and secure placement of boxes onto the feeding lines, as well as the simulation with the manipulator to validate the developed models.

The planning of trajectories for the redundant arm consists of executing various manipulation tasks. These tasks involve picking up and placing *Targets* between the storage buffer and the autonomous mobile robot (AMR) and, in turn, picking up and placing these boxes onto the access ramps. To fulfill this primary goal, a series of preliminary tasks must be undertaken, such as familiarizing with the Robot Operating System (ROS) and MoveIt frameworks. The creation of a simulation scenario replicating the real environment at Bosch Car Multimedia BrgP using the physical simulator CoppeliaSim as well as determining the number of boxes the AMR can transport across the shop floor are fundamental to box manipulation management.

Another important subtask to consider is the study of possible modifications in the “as is” shop floor, particularly the access ramps, which in the current scenario reveal difficulties for the placement of the *Targets*. Alongside these suggested changes, this research will also extend to the search for an appropriate gripper in the market for the robotic arm. The gripper must be capable of grasping the boxes for manipulation while satisfying the project's requirements.

To ensure the robotic arm's movements are legible and safe, a comprehensive overhaul of the collision-free trajectory planners will be conducted. A comparative analysis of these planners will be made to select the most proper one for the pick-and-place operations.

This dissertation also aims to settle a means of communication between MoveIt, the motion planner library responsible for projecting the robot's movements, and the physical simulator CoppeliaSim, which will transmit all the physical interactions and overall operations.

As expected outcomes, the goal is to achieve a robotic arm capable of performing safe and human-readable movements during pick-and-place operations of boxes in the production

area. This is particularly important as the robot will coexist in an environment with human operators, where human-robot interaction must be as efficient as possible.

1.3 Structure of the Dissertation

This dissertation is divided into six chapters, each explained below. The current chapter serves as the introduction, addressing the problem definition, providing context for the MIAR project, and outlining the motivations and goals established for this work.

The second chapter exposes the current growth in the use of robots in the industry, focusing on the evolution of collaborative robots in shared tasks. It also includes a general review of collision-free motion planners, especially sampled-based, optimization-based, and human-like planners. It addresses an analysis of the industry's existing automatic feeding and collection systems for boxes, and it exhibits an overview of solutions similar to the robotic approach presented in the MIAR project, emphasizing control by motion planning for mobile platforms and integrated robotic arms.

The third chapter explores theoretical foundations important for properly understanding the developed work and delves into the impact of redundancy in the robotic arm's kinematics study. It also lays out the planners under examination in the project for the pick-and-place trajectories. Lastly, it details all the software and hardware necessary to realize the respective work.

The fourth chapter represents the core of this dissertation. It describes the system architecture of the MIAR project and details all the practical tasks carried out during the project, commencing with the creation of the simulation scenario in CoppeliaSim, the study of the number of boxes the AMR can dislocate, and the selection of an appropriate gripper for the manipulation tasks. It includes the modeling and simulation of potential gripper fingers to be used in the project. The design of optimal access ramps to facilitate pick-and-place operations for the *Targets* is also examined. Other tasks were also executed, such as configuring the manipulator with the gripper in MoveIt, replicating the robot's workspace environment to the Rviz, and establishing communication between MoveIt and CoppeliaSim.

The fifth chapter thoroughly compares the motion planners introduced in Chapter 3 to identify the most suitable one for the project. Consequently, it includes the results and

analysis of the benchmarking study. It also explores customized planners and examines how their adjustments influence the performance of the pick-and-place operations.

The final chapter provides a comprehensive summary of the work accomplished throughout this research, highlighting the conclusions drawn from the developed solution. It discusses the possible limitations of the proposed approach and outlines suggestions for future research.

The dissertation concludes with an annex and appendices that include technical drawings of the access ramps, the fingers designed for the selected gripper, and the results of the motion planners comparison.

2 State of the Art

This chapter begins by discussing the evolution of collaborative robots in the industry and provides a literature review on automated solutions for the collection and placement of boxes on production lines. It also summarizes motion planning methodologies that ensure collision-free movement, categorizing them into global and local planners. Finally, the chapter highlights some applications of these methodologies in mobile manipulator robots.

2.1 The Importance of Collaborative Robots in Industry

Industry 4.0 represents a major transformation in industrial processes. This digital industry relies on cyber-physical systems and the Internet of Things to connect databases between systems and equipment, such as sensors and processors, to promote a safer and more efficient industry (Toktaş-Palut, 2022).

In modern industry, robots are the fastest-expanded mechanical solution implemented in the industrial sector (S. Liu & Liu, 2020). Based on the World Robotics 2023 report, the worldwide annual installation of industrial robots has tripled from 2012 to 2022, presenting a slight increase of 5% from 2021 to 2022. This growth can be checked in Figure 1 – (a) graph. Besides, between 2021 and 2022, there was a notable increase in the annual Installations of Industrial robots by customer industry, mainly in the electrical/electronics sector of 10% and in the automotive sector of 16%, as depicted in Figure 1 – (b).

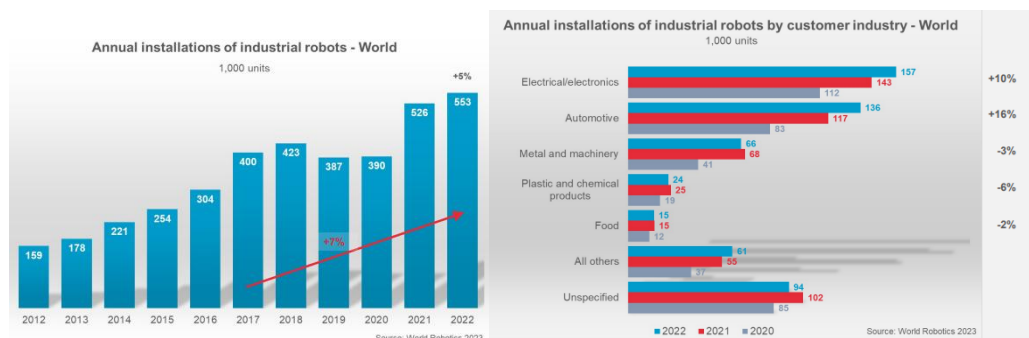


Figure 1 - (a) Annual installations of Industrial robots - World; (b) Annual Installations of Industrial robots by customer industry – World. Adapted from (IFR International Federation of Robotics, 2023).

Companies adopt more and more industrial robots for their extensive usability in operational processes such as manufacturing and logistics, especially robotic arms, to execute repetitive, exhausting, and dangerous tasks like arc and spot welding, carrying high loads, solving manual labor problems, and ensuring higher production efficiency. These types of robots are usually in a robotic cell equipped with sensors and protective fences to surround the robot's workspace to ensure the safety of human workers (Weidemann et al., 2023).

Industrial robots have been defined in different ways over the years. The Robot Institute of America (1979) states that an industrial robot is "a reprogrammable, multifunctional manipulator designed to components or specialized devices, through programmed motions for the performance of a variety of tasks" (Considine & Considine, 1986). According to the international standard EN ISO 10218-1:2011 (International Organization for Standardization, 2011a), which specifies safety requirements for industrial robots, an industrial robot is "an automatically controlled, reprogrammable, multipurpose manipulator, programmable in three or more axes, which can be either fixed in place or mobile for use in industrial automation applications."

Despite the continued growth of traditional robots, there is an increasing demand for more dynamic work environments where tasks are less structured, and workers can collaborate alongside robots (Palmieri & Scoccia, 2021). In these cases, the robots don't have protective fences and work aided by safety sensors and image processing through cameras. Consequently, the study and development of collaborative robots have expanded.

Collaborative robots, also known as cobots, emerged in the 1990s. A collaborative robot is defined as a robot designed for direct interaction with a human within a defined collaborative workspace (ISO 10218-2:2011 (International Organization for Standardization, 2011b)).

In contrast to traditional robots, the purpose of cobots is to work alongside humans in a variety of industrial applications like palletizing, machine tending, assembly, and injection molding, as well as in other areas such as healthcare and assistance for people in need. Cobots have been steadily evolving over the years to make collaboration with humans more efficient and safer and to enable improvements in workplace ergonomics. They will primarily be used for repetitive tasks that require high precision and load-lifting capabilities (Weidemann et al., 2023).

The cooperative interaction between robots and humans in the same environment is described as Human-Robot Collaboration (HRC). The collaboration of those agents aims to enhance work performance by combining robot strengths with human cognitive skills, thereby reducing the physical effort required from human operators.

While traditional industrial robots are immobile and heavy (weight > 50 KG), feature long service life, require large operation space, and work with high loads and working speeds, collaborative robots are typically more flexible, lightweight, embedded with security features, can be mobile, have low payload capacity, low velocities, and simple setting, not affecting layouts in factory floors (Matúšová et al., 2019). Cobots are also easier to program and teach, especially with the deployment of intelligent technologies like 5G technology and artificial intelligence.

Because of these advantages, the number of cobots in use within the industry has been increasing over the years (Figure 2).

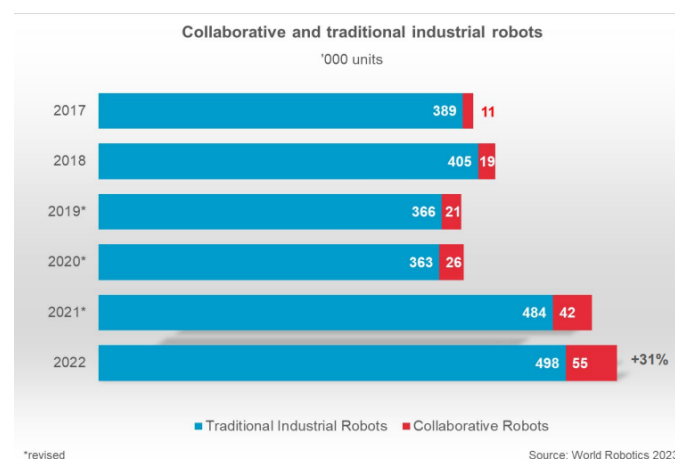


Figure 2 - Installations per year of collaborative and traditional industrial robots, 2017 to 2022. Adapted from (IFR International Federation of Robotics, 2023).

Despite the benefits of collaborative robots, their integration into industrial environments typically requires technical robotics expertise, which can limit their acceptance within the sector (Matheson et al., 2019).

In order to implement these robots in the workplace, they must comply not only with the industrial robot safety standards (ISO 10218) but also with the specific requirements for collaborative robots, such as maximum work speed (ISO/TS 15066:2016 (International Organization for Standardization, 2016)). To ensure safety, cobots must have one of the

following safety modes: monitored stop when humans are nearby, hand guidance for operations controlled by humans, speed variation based on proximity areas, and limited power and force to reduce injury risk during potential collisions (Guertler et al., 2023).

In collaborative operations, the robot and the human operator are active agents. The robot's speed and motion can affect the human worker's performance. If the robot's motion is unexpected and fast, it can create anxiety and discomfort for the human co-worker. Therefore, cobots tend to have an anthropomorphic appearance and, instead of working with functional motion, like traditional robots, must be predictable and legible. While a predictable motion aligns with the human observer's expectation of the action of the cobot, a legible motion describes the intention of the robot's action through its movement. A study presented by Dragan (2015) revealed that legible motion generally contributes to more efficient collaborations than predictable and functional motions. This leads to the concept of Industry 5.0 (Doyle-Kent & Kopacek, 2021), where humans and robots work side by side on the same tasks. While Industry 4.0 is primarily focused on ensuring quality, data collection, and data flow, where humans and robots, though they may work on the same production line, operate independently, Industry 5.0 is focused on satisfying the needs of human workers with the assistance of robots that coexist in the same work environment.

2.2 Automatic Systems for Feeding and Collecting Boxes from Production Lines

Nowadays, the production lines are becoming more and more automated. The current growth of automatic systems for feeding and collecting packages from production lines has brought out several benefits, such as high operational efficiency, less likelihood of human error, lower workplace injuries, and flexible services (Azadeh et al., 2019). Storage buffers are areas where the products are temporarily stored before being reinserted into the production lines or sent to other sectors. Over the years, many developed automated systems have collected and returned products to buffers. This section will present some of these solutions.

Automated Storage and Retrieval System (AS/RS) is a warehousing system that manipulates, stores, and retrieves products with precision under a defined degree of automation (Roodbergen & Vis, 2009). Thus, an AS/RS involves a series of computer-controlled processes to place and retrieve loads from predefined storage areas automatically.

This type of system comprises the warehouse structure, the storage modules, which are the shelves that store the loads and are arranged in rows, and the storage and retrieval (S/R) machines. The S/R machines are responsible for picking and placing loads between the input/output stations and the storage area, moving the loads horizontally and vertically along the rows using elevation systems. The S/R machines commonly used in AS/RS are stacker cranes. AS/RS is also formed by control modules, which use computers and programmable logic controllers to guide the S/R machines to the desired positions, and Pick-up and drop-off (P/D) stations, which transfer the materials in and out of the AS/RS (Xing et al., 2010). Figure 3 illustrates an AS/R system and its components.

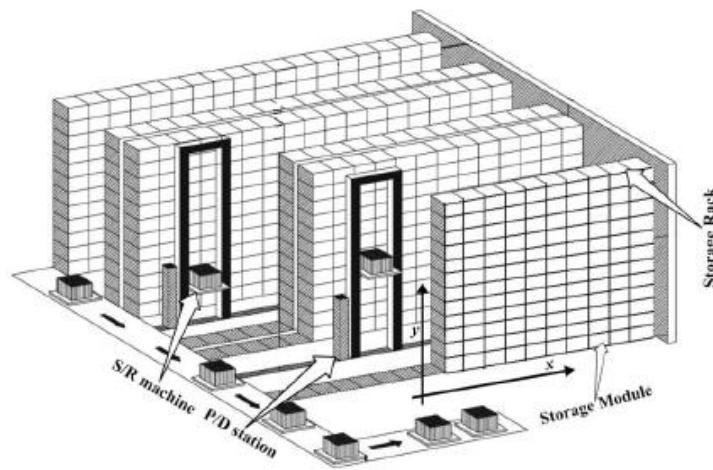


Figure 3 - Example of an AS/RS layout. Adapted from (Xing et al., 2010).

This system offers numerous benefits, including increased storage capacity, enhanced productivity, reliable handling, and reduced need for human intervention (Groover, 2007). However, it presents significant implementation and maintenance costs (Peixoto, 2015).

AutoStore is a robot-based compact storage and retrieval system (AS/RS) fully operated by robots above a structural grid. This order-picking system stores small parts in storage bins that are stacked over each other (Trost et al., 2022), following the 'Last-In-First-Out' storage strategy, ensuring better use of the available space in other storage systems. This system can be seen in Figure 4.

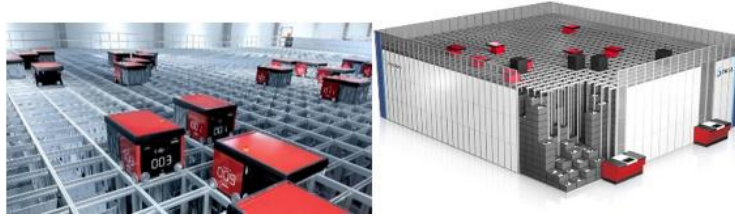


Figure 4 - Example of an AutoStore system. Adapted from (Trost et al., 2022).

The grid is an aluminum or steel structure that serves as a rail network for the robots and keeps the columns of containers stacked. In this system, the robots carry out the process of storing an SKU (Stock-Keeping Unit) completely autonomously. The goods are placed in a container at the picking station, and a robot will lift the container to the grid. Usually, the bins with high-turnover products are positioned at the top of a stack in the grid for quicker access and retrieval, while bins with lower-demand SKUs are migrated to the bottom of the stack. If an SKU is needed for retrieval, a robot is sent to find the given bin. The retrieval process is direct if the bin is located at the top of a stack, as the robot picks the bin and transports it to the picking station. However, if the access to the given bin isn't direct, it will be necessary to relocate all the containers that are stacked above the desired bin. Once a robot retrieves the bin and the goods inside the bin are removed to the station, another robot on the top of the grid relocates the containers that were moved back to their original stack order (Trost et al., 2022).

Autonomous Mobile Robots are reshaping warehousing operations by taking over material handling processes. These flexible systems are intelligent, developed to navigate alongside human workers, and equipped with many components such as onboard sensors and cameras. AMRs move autonomously without an operator, reducing repetitive tasks and labor costs, minimizing the likelihood of workplace accidents, and enhancing workflow and logistics management (Niermann et al., 2023).

Mobile manipulators, like the solution used in the MIAR project, are an AMR solution where a robotic manipulator is integrated into the mobile base. This system is increasingly used in industrial and warehouse environments in operations like machine tending and inspection of large components (Annem et al., 2019). The application of pick-and-transport tasks using a mobile manipulator can be seen in the case study presented by Thakar et al. (2022) in Figure 5. In this approach, the mobile robot transports the parts to the intended stations in a cluttered environment while the manipulator picks and places the parts in the

intended locations. The motion of the mobile base and the robot arm movements for grasping and releasing are handled separately.

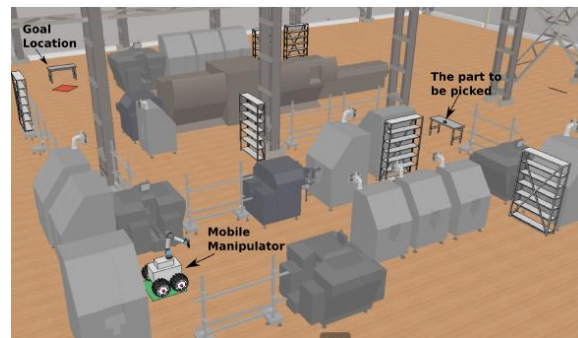


Figure 5 - Example of a warehouse scenario where a mobile manipulator performs pick and place tasks. Adapted from (Thakar et al., 2022).

Another robotic solution for feeding and transporting bins consists of a mobile robot with fixed rack storage, such as the Autonomous Case-handling Robotic System (ACR) from HAI ROBOTICS (Hai Robotics, 2022). This flexible system is an automated, unmanned robot designed for picking and handling tasks. The AMR operates as a good-to-person (GTP) solution by using automated transportation of goods, handling and moving multiple boxes simultaneously, improving picking efficiency and storage capacity while optimizing the time for storage and retrieval tasks. This solution features a lift system to load and unload the goods. It consists of a mast supported with several carriers/shelves for storing bins and a load-handling unit that can rotate horizontally and move along the mast to grab goods at different heights in the racking locations (Figure 6). To place goods, the AMR takes the grabbed bins to available storage locations within the racking and uses its arms to push the bins onto the shelves. To retrieve the goods, the AMR moves to the appropriate storage rack, the load-handling unit is lifted to the intended storage level, and using its arms, the bin is grabbed and placed in one of the empty carriers of the mast.



Figure 6 - Autonomous Case-handling Robotic System (ACR) from HAI ROBOTICS. Adapted from (H. Robotics, 2022).

Similar solutions to those previously presented are the Skyler and Cart robots from Caja Robotics (C. Robotics, 2023). These flexible good-to-person systems typically work together to enhance order-picking operations by moving bins between workstations and shelves. The main difference between these solutions and those mentioned earlier is that these do not contain carriers on the mobile robot. They are only composed of the load-handling unit. The Cart Robot is designed for tasks requiring speed and is used for picking and inventory validation, while the Skyler robot is used for high-height (Figure 7). Moreover, both robots can handle objects measuring up to 403 x 610 mm and are equipped with safety sensors for safe navigation.



Figure 7 - Skyler and Cart robots from Caja Robotics. Adapted from (C. Robotics, 2023).

2.3 Motion Planners

Robotic arms are particularly suited for processes that involve picking and placing components, where timing and precision are critical. For a robotic arm to successfully transport boxes, as required for the MIAR Project, it is essential to plan its movements carefully to avoid collisions with obstacles in the working area and with the robot itself. As manipulators are increasingly required for challenging operations, often working in tight spaces and collaborating with humans, the importance of effective motion planning continues to grow. Motion planning has been a subject thoroughly explored for years (LaValle, 2006), which consists of determining a robot's motion from an initial state to a target state, avoiding environmental obstacles, and satisfying constraints, such as joint and torque limits (Latombe, 1991). The motion planning algorithms can be categorized into:

- **Classical (Deterministic/Combinatorial) Methods:** These methods define an explicit representation of the topology of the configuration space. They are rigorous in discretizing the configuration space and searching for the best path, often leading to higher computational demands as the dimensionality of the configuration space increases. The combinatorial methods are decomposed by the cell decomposition, potential fields, and roadmap approaches (Choset et al., 2005).
- **Grid Methods:** These approaches discretize the configuration space into a grid structure, facilitating searches for viable paths (Maria & Rodrigues, 2022).
- **Sampling-Based Methods:** These techniques utilize deterministic or random sampling to explore the configuration space, making them particularly adept at handling high-dimensional problems.
- **Optimization-Based Methods:** These methods aim to generate optimal trajectories based on specific criteria, such as minimizing path length, motor efforts, or computation time. Optimization-based methods can produce efficient and smooth paths but might require significant computational power.
- **Human-Like Motion Planning Methods:** With the increasing integration of robots into shared workspaces alongside humans, more and more motion planning algorithms prioritize legibility and predictability. These methods aim to generate intuitive movements for humans, enhancing safety and collaboration (Gulletta et al., 2021).

However, motion planning algorithms are typically classified into two types of path-planning methods for obstacle avoidance: the global and local methods (Hwang & Ahuja, 1992). The following sections will discuss the concept of each method and provide an overview of some existing motion planning algorithms for each.

2.3.1 Global Planners

The global motion algorithms (Hwang & Ahuja, 1992) consist of methods that process information about the robot's workspace before executing the movement. These methods are used when the motion planning must be meticulously planned. The process in this planner is made offline. An example is planning motion trajectories for industrial robots in environments where avoiding obstacles is crucial. In the global approach, since the workspace is static, the motion planning for robotic arms becomes easier, resulting in generated movements that are more faithful to human movements and providing more predictable actions to the human operators around the robot's workspace. Some motion planning algorithms classified as global methods, such as sampling-based planners and optimization-based Planners, are presented below.

2.3.1.1 Sampling-based Planners

Sampling-based motion planning has been one of the most researched path-planning methods in recent years. These methods are widely adopted in high-dimensional spaces (Hsu et al., 1997). The sample-based methods also referred to as probabilistic methods, use random or deterministic sampling functions to explore the configuration space (C-space). The main purpose of these planners is to solve path-planning problems by constructing graphs or trees through the connection of sampled configurations (Choset et al., 2005). Therefore, these planners can be divided into two main phases: the first phase involves sampling the configuration space to create a roadmap, and the second is searching this roadmap to find an optimal path (Barland, 2012). Among all types of planners, the most popular are the Probabilistic Roadmap (PRM) and the Rapidly Exploring Random Tree (RRT) (Elbanhawi & Simic, 2014).

2.3.1.1.1 Probabilistic Roadmap Method

The Probabilistic Roadmap Method (PRM), introduced by Kavraki et al. (1996), is a planning method that searches for free collision paths for robotic manipulators with many degrees of freedom (DoF), known for effectively solving complex motion planning problems.

This method involves creating a roadmap (a graph) in the configuration space and searching for the optimal path. Thereby, the algorithm comprises two distinct phases: the learning phase and the query phase. In the learning phase, the algorithm constructs a probabilistic roadmap by continuously generating numerous random configurations within the robot's Collision-free space (Cfree). These collision-free configurations are represented as nodes in a graph, and due to a local planner, free collision paths are calculated by binding the nearby configurations (nodes) together, generating the edges of the roadmap. The query phase consists of finding the most feasible path between a specified initial configuration (q_{init}) and a final configuration (q_{goal}) on the roadmap created in the learning phase. The method attempts to find the nodes on the roadmap closest to the nodes representing the initial and final configurations and then establishes a connection between those nodes. Subsequently, a path-finding algorithm is performed to find the shortest path between the configurations. An advantage of the PRM algorithm is the ability to handle many types of robots with arbitrary DoF, and it doesn't require detailed information about the robot's configuration space. It only requires the ability to verify whether a given configuration is collision-free. Figure 8 shows an example of the construction of a roadmap.

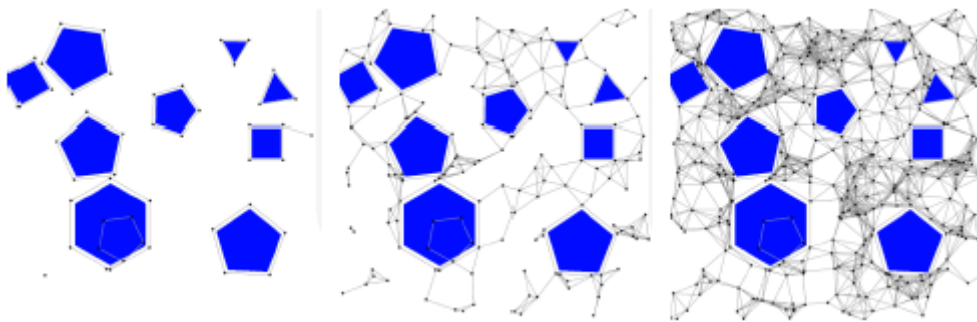


Figure 8 - Example of building a roadmap on a robot. Obstacles are represented in blue. Adapted from (Eric O. Scott, 2015).

There have been several extensions to the PRM planner over the years, such as the Lazy PRM algorithm developed by Bohlin & Kavraki (2000). In this method, the collision checking is only carried out after a path between the initial and final configurations is found, significantly reducing computational time in many scenarios. To do this, this approach initially considers that all the roadmap nodes and edges are collision-free before searching for the shortest path, and only afterward are the nodes and edges checked for collisions.

2.3.1.1.2 Rapidly Exploring Random Tree

The Rapidly Exploring Random Tree (RRT) developed by LaValle (1998) was designed to solve motion planning problems with algebraic and differential constraints in high-dimensional spaces. This probabilistic method creates a graph in the shape of a tree that is expanded rapidly within the robot's configuration space. To implement this method, defining the environment, the initial and goal points, and the maximum number of samples is necessary. The tree is rooted from the initial state and grows incrementally by selecting and connecting random points until it either reaches the goal position, the number of iterations, or the planning time expires (Kang et al., 2019). A new point, corresponding to a new robot state, is randomly generated for each iteration. The nearest point to the tree is searched, and a local planner attempts to extend the tree to the sampled point. If the new point is collision-free, it is added to the tree, creating a new branch. When the algorithm is unable to find a solution in some instances, the problem can be addressed by executing the algorithm with more samples. Unlike the PRM, which requires the construction of a roadmap and the connection of nearby configurations, in the RRT, the samples are obtained uniformly, causing the tree to grow into regions of the collision-free space that have not yet been explored. As a result, RRT is widely used in high-dimensional spaces and nonholonomic and kinematic motion planning tasks (Gasparetto & Zanotto, 2010). The RRT planner is often faster than the PRM planner because it focuses on finding the nearest neighbor point for each iteration. At the same time, the PRM searches through multiple neighbor points around a random sample. Figure 9 illustrates an example of the construction of a configuration tree.

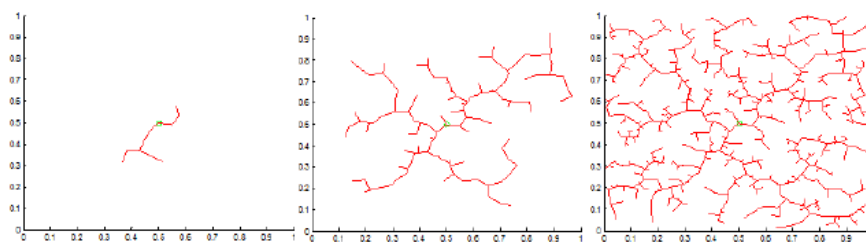


Figure 9 - Example of the building of an RRT tree. Adapted from (Barland, 2012).

2.3.1.1.3 RRT-Connect

The RRT-Connect algorithm, developed by Lavalle & Kuffner (2000), also known as bidirectional RRT, was designed for efficient single-query path planning. This approach rapidly

finds solutions by exploring the configuration space with two random trees that grow toward each other with a simple heuristic. The algorithm starts by creating two trees: one “forward” tree rooted in the initial robot configuration and the other “backward” that starts at the goal configuration. In this single-query probabilistic planner (B. Sciavicco & Oriolo, 2009) the trees expand incrementally by selecting a sample of the C_{free} . One of the trees then reaches out toward the sample, creating a new configuration. The algorithm attempts to connect the configuration to the nearest vertex of the other tree through a “Connect” function that iteratively tries to link the two trees. If an obstacle is found in the connection, the trees switch roles, and the algorithm starts the next iteration. RRT-Connect is considered probabilistically complete, which means that the probability of finding a solution increases with longer planning time. It is also usually faster than the standard RRT, especially in less cluttered environments, mainly because it focuses on connecting the two trees, while RRT explores with a single tree until it reaches the goal state.

2.3.1.1.4 RRT*

The Rapidly Exploring Random Tree Star (RRT*) algorithm developed by Karaman & Frazzoli (2011) is a variation of the single-tree RRT algorithm designed to ensure feasible and optimal paths according to a specified cost function. RRT* addresses the asymptotic optimality, which means that as number of samples increases, the probability of finding the optimal path approaches one. This algorithm modifies the RRT by introducing a process that continuously rewires the search tree. For each iteration, the algorithm starts by considering that the minimal cost connection will come from the closest sample to the tree. Therefore, a new sample point is added to the tree, and the connections of nearby configurations are re-evaluated to ensure that they contribute to the most cost-effective path. This iterative process is continuously optimized until an optimal path is found. Although the RRT* provides significantly better solutions, it requires increased computational cost since the process of rewiring the tree demands additional calculations, leading to longer computational times and a slower convergence rate. Nevertheless, adding asymptotic optimality in this sampling-based method increases the time complexity by only a constant factor over the standard RRT and still provides better paths.

2.3.1.2 Optimization-based Planners

Optimization-based planners address motion planning tasks as optimization problems, where the goal is to compute the most efficient trajectory that satisfies a set of constraints

related to the robot. To find an optimal trajectory, an object function must be defined, with the user selecting what parameters wants to optimize, such as time, energy, or other factors (Barland, 2012). These planners often use either gradient-based methods, which optimize by incrementally adjusting the trajectory, or non-gradient methods, which explore a broader range of possible solutions (Lynch & Park, 2017). Although these methods are widely used due to their capacity to generate trajectories that optimize cost, these approaches don't always ensure an effective solution. Among the several trajectory optimization approaches, some techniques stand out.

2.3.1.2.1 Covariant Hamiltonian Optimization for Motion Planning

Zucker et al. (2013) introduced an optimization-based motion planner named Covariant Hamiltonian Optimization for Motion Planning (CHOMP) algorithm that uses covariant gradient and functional gradient techniques to optimize robot trajectories for smoothness and obstacle avoidance. This approach is based on the potential field work of Quinlan & Khatib (1993). Unlike most motion planners that treat planning and optimization as separate stages in trajectory generation, CHOMP merges these phases through gradient-based optimization techniques to create an entirely optimizing trajectory. CHOMP models a trajectory as a spring-mass system, where the path internal energy is related to the length and smoothness, and the external energy is related to the obstacles. This algorithm seeks to identify the most efficient path that minimizes the total energy of the trajectory. The trajectory's cost is defined by two components: the smoothness functional of the planner, which measures the dynamics quantities of the trajectory, which aims to reduce quantities like the squared velocity norm during the execution time, and the obstacle-avoidance functional that implements cost functions in the robot's operational space to keep the algorithm complexity low (Gulletta et al., 2021). The planner quickly adjusts the trajectory to avoid collisions while optimizing dynamic quantities like the robot's joint velocities and accelerations.

2.3.1.2.2 Stochastic Trajectory Optimization for Motion Planning

Another optimization-based motion planner, known as Stochastic Trajectory Optimization for Motion Planning (STOMP), was developed by Kalakrishnan et al. (2011). This algorithm was designed for robotic arms to generate smoothness trajectories that avoid obstacles and minimize costs related to constraints in a given time. Unlike gradient optimization methods like CHOMP, STOMP doesn't require gradients for the planning. As

mentioned, it optimizes general costs like torque limits, energy, and constraints included in cost functions. This planner starts by randomly generating, at each iteration, a set of noisy trajectories around an infeasible initial trajectory through a normal probability distribution. These noisy trajectories are evaluated based on a cost function that includes terms like trajectory smoothness and obstacle avoidance. The results from the evaluation are used to iteratively update to a trajectory with lower cost. The higher-cost trajectories contribute less to the final optimized path. It is considered that both initial and final arm postures in the trajectory are permanent during the optimizing process, and the duration of a given trajectory is fixed. STOMP benefits from its ability to quickly compute distance queries and collision costs using distance fields and spherical approximations. Since STOMP is not a gradient-based approach, it has the advantage of overcoming local minima issues, a common problem in CHOMP. Furthermore, practical experiments revealed that STOMP presents a better success rate in various manipulation operations than CHOMP.

2.3.1.3 Human-like Planners

There is an increasing interest in ensuring that robots collaborate with humans on shared tasks. Human-Robot Interaction/Collaboration (HRI-C) is enhanced when robots possess an anthropomorphic appearance and exhibit movements similar to those of humans. This makes their actions less unpredictable and aids human perception in understanding robotic behavior. In other words, humans can intuitively follow a robot's movements and anticipate its future actions (Gulletta et al., 2020). Even though probabilistic-based approaches have been extensively explored and studied over the past decade, these planners usually produce unpredictable trajectories (Rafael & Ferreira, 2022). Unstable movements, such as sudden operating speeds by robots, can cause anxiety among nearby humans and pose dangers in shared environments (Zacharias et al., 2011). Human-like morphology and movement are key characteristics for effective human-robot interactions. Robotic movements should be perceived as natural, predictable, and capable of demonstrating the underlying intent of the action without the need for verbal communication (Gulletta et al., 2021). Therefore, the study of legible and smooth robotic movements has led to the development of various human-like approaches inspired by the human motor principles presented below.

2.3.1.3.1 RULA Criterion with RRT-Connect

Zacharias et al. (2011) developed a global method for redundant robots using the RULA (Rapid Upper Limb Assessment) ergonomic research criterion to generate human-like

movements. In this method, the criterion is applied to identify the areas in the workspace of a manipulator where it exhibits configurations similar to those of a human. RULA measures stress and strain levels in human upper limb postures on a 1 to 7 scale. The lower the RULA score, the more natural the robot configuration is (McAtamney & Corlett, 1993). The initial and desired configurations of the arm are determined through inverse kinematics guided by RULA. Subsequently, the trajectory to move the arm between the two configurations is generated using the RRT-Connect sampling algorithm, thereby defining a collision-free trajectory. To find the natural configurations, the robot's dexterous workspace is composed of regions with a reachability index for measuring the success in finding inverse kinematics solutions. Thereafter, the RULA criterion is applied to this map, where points with high RULA scores are excluded, indicating less natural postures. A drawback of this method is that if we set a lower RULA score requirement for planning, we will have fewer valid manipulator configurations, resulting in fewer inverse kinematics solutions. Additionally, this method also overlooks the typical temporal parameterization seen in human movements, leading to an unrealistic planning time for the robot.

2.3.1.3.2 Bi-RRT Algorithm with TAPs

Xie et al. (2011) introduced the TAP "Target Arm Pose" concept for planning human-like arm movements in robotic manipulators. Before starting a task, humans project their arm's final pose, called TAP. TAPs are defined through a qualitative assessment using performance indices that compare the robotic arm's configuration with the human arm's pose in specific target positions. In a collision-free path, a minimum jerk model is employed to generate smooth movements. To apply the model, the initial posture of the robotic arm, the start point, the endpoint, and a set of waypoints must be known. Next, the minimum jerk model creates a path for the finger through all these points. TAPs are calculated at each waypoint and target position to ensure the robot's movement patterns mimic human arms. The IK-JIL inverse kinematics algorithm enables the end-effector to follow the finger path while reaching the designated TAPs. In a scenario with obstacles, a global space analysis identifies high-risk collision areas. Crucial points are selected to create TAPs for the arm that avoid those obstacles. The arm's path is segmented, and the Bi-RRT algorithm is applied to each segment to determine a collision-free path. A disadvantage of this solution is the lack of consideration for typical human motion velocity profiles, such as the bell-shaped velocity profile (Gulletta et al., 2021).

2.3.1.3.3 Analytical Inverse Kinematics Solver for Anthropomorphic 7-DOF Redundant Manipulators with Human-Like Configuration Constraints

W. Liu et al. (2017) proposed an analytical inverse kinematics algorithm for seven degrees of freedom robot manipulators with shoulder-elbow-wrist (SEW) configuration. This algorithm is designed to replicate human configurations from recorded human movements. It introduced the concept of key positions, which are cartesian positions of the joints of the manipulator. A correspondence method named wrist-elbow-in-line allows mapping key positions from human demonstrations to the manipulator. The mentioned key positions correspond to the shoulder, elbow, wrist, and end-effector/hand (Figure 10 – (a)). Only the key positions of the manipulator's elbow and wrist joints are used as constraints to define the robot arm configuration since the key position of the shoulder is known because it's fixed, and the task determines the end-effector. In this method, the key position of the wrist must be aligned with the manipulator's elbow and end-effector key positions, ensuring all remain within the same reference plane. Additionally, the key position of the manipulator's elbow must be as close as possible to the human elbow's key position (Figure 10 – (b)). When mapping the key positions from the human demonstration to the robot, the robot's joint limits and link lengths must be considered. Therefore, this solution allows the robot's elbow key position to be the closest to the human's elbow key position by adjusting iteratively the robot's wrist key position until a valid analytical inverse kinematics solution is found. A disadvantage of this solution lies precisely in solving the correspondence problem. As a strategy of iterating key positions is applied, the robot's configuration ceases to follow a human-like posture in areas near the joint limits of the robotic arm.

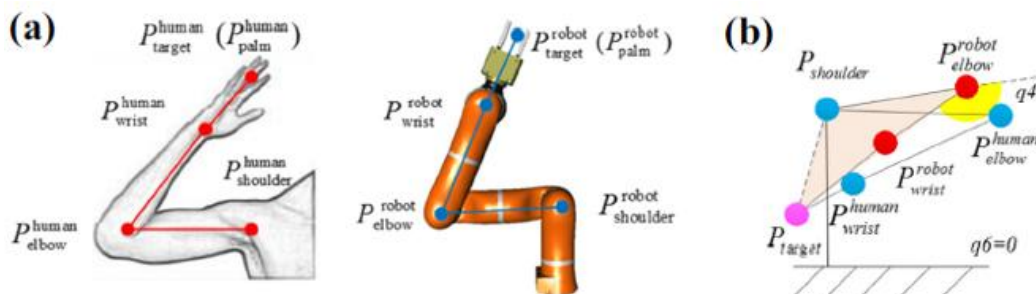


Figure 10 - (a) Kinematic structure of a robot and human arm and their respective key positions; (b) Correspondence method wrist-elbow-in-line. Adapted from (W. Liu et al., 2017).

2.3.1.3.4 Human-like Upper-limb Motion Planner

Gulletta et al. (2021) proposed a motion planning algorithm named HUMP - Human-like Upper-limb Motion Planner. This planner was implemented for point-to-point movements and pick-and-place operations, does not require a high computational cost, and contemplates biological temporal parameterization through Fitts' Law. In this global method, trajectories are pre-planned offline to avoid collisions during the robot's path before starting its movements through sensory input information from the workspace. To provide a smooth and human-like motion, the target posture (TP), which corresponds to the robot configuration that places the robot's end-effector in the intended pose, must be chosen to minimize the angular jerk of the movement between the robot's start posture (SP) and the TP. This movement in joints-space is referred to as direct movement (Figure 11 – (a)). If obstacles are anticipated in the direct movement, bounce Postures (BP) are selected to facilitate a back-and-forth movement, which involves moving from SP to BP and back to SP. In this way, this algorithm considers the direct movement and back-and-forth movement, which are used on paths with obstacles. When these two movements are combined, a collision-free trajectory is defined (Figure 11 – (b)). In the proposed method, the movement was composed of pick, place, and move segments (Figure 11 – (c)). The pick and place movements combine three phases: transport, approach, and retreat. The approach phase corresponds to the moment when the manipulator moves closer to grasp the object in the pick segment and when the object is placed in the desired position in the place segment. Meanwhile, the retreat phase corresponds to the moment when the object is lifted in the pick segment and when the manipulator moves away from the placement position in the place segment. The move segment includes the transport phase, which moves the gripper toward the desired positions. The approach and retreat phases are linked by the actions of grasping and ungrasping the manipulated object. All these phases require the selection of a TP and the generation of direct movement. The transport phase is the only one that involves bounce postures to generate back-and-forth movements and subsequently create a composite movement, as it is the only phase that considers objects along the path.

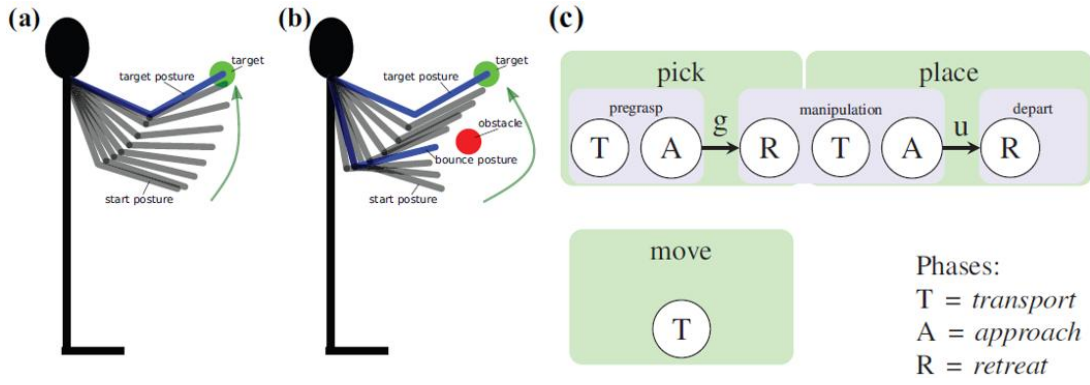


Figure 11 - (a) Direct movement from start to target; (b) Composite movement; (c) Division of the different phases of the pick and place movement. Adapted from (Gulletta et al., 2021).

2.3.2 Local Planners

The local methods plan the movement online, considering only small portions of the workspace obtained immediately. The trajectory is adapted in real-time, making these methods suitable for unstable or dynamic cases where the environment may change suddenly, requiring continuous trajectory adjustments (da Costa, 2019). Tasks like serving a glass of water to a person at a table require this method most of the time due to unpredictable changes in the scenario and the uncertainties inherent in the task itself. However, in the local approach, since the robot operates in a dynamical environment, due to the real-time collision avoidance, the generated movements lose fidelity, making it challenging to meet requirements such as kinematic characteristics similar to those observed in human movements (Gulletta et al., 2021). Often, the local planner is combined with a global planner as a safety measure to avoid unexpected objects that appear in a planned trajectory. The global planner generates an initial solution or iteratively optimized solutions during the execution, while the local planner is running continuously during the execution (Vemula et al., 2021). Local planners are crucial, especially in industrial automation, requiring fast responses. These planners stand out in tasks where the robotic arm must adjust to unpredictable changes and prioritize real-time trajectory adjustments to ensure safety. Therefore, they are widely used in collaborative operations with human operators. The following section will address some of the most well-known local motion planning algorithms.

2.3.2.1 Potential Field Approach

Khatib (1986) introduced the first local method for motion planning, known as the potential field method, which is a simple yet effective approach for real-time applications. In

this method, the robot is treated as a point particle, and the robot's configuration space is explored and modeled as an artificial field of forces, where the goal configuration has a negative virtual potential that exerts an attractive force on the robot and the obstacles around possess a positive virtual potential which exerts repulsive forces. The sum of the virtual potentials represents the total potential field. Hence, the robot is repelled from the obstacles, ensuring that it avoids them during the trajectory, and it is attracted to the final point, creating a collision-free path between the initial and goal configuration (Hwang & Ahuja, 1992). A simple example of how this planner works can be seen in Figure 12.

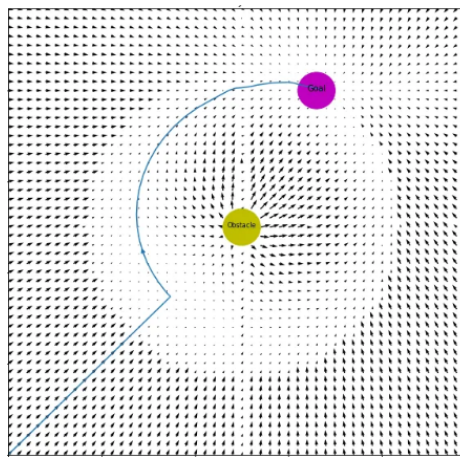


Figure 12 - Example of a path taken by a robot using the potential field method. Adapted from (Aslan, 2021).

The potential field method is computationally efficient and widely used in robotic applications. It allows quick real-time adaptation of the robot's path without prior knowledge of the configuration space and without pre-processing to compute the whole path a priori. A drawback of this method is that the robot can become stuck in local minima. This happens when the attractive and repulsive forces in the potential field have the same potential and balance out, resulting in a zero gradient that can trap the robot in a non-goal position. Several strategies, such as discretizing the configuration space (B. Sciavicco & Oriolo, 2009) have been studied to solve this problem.

2.3.2.2 Attractor Dynamics Approach

Attractor Dynamics is another local method for motion planning that is useful in dynamic and unpredictable environments, especially in shared workspaces such as operations involving human-robot interaction (Iossifidis & Schoner, 2004). The attractor dynamics method was originally developed as a variant of the potential fields approach. In this

approach, the dynamical systems are defined by attractor states, which represent the desired states and act as attractive forces guiding the system toward the goal, while the obstacles act as repulsive vector fields, ensuring collision-free trajectories. Unlike traditional potential fields, on attractor dynamics, motion planning is generated using the direction of movement rather than just the cartesian position of the robot. The direction of movement comprises two heading direction angles that determine the end-effector's trajectory in three-dimensional space and the end-effector velocity vector. Rano & Iossifidis (2013) created a human-like motion method based on the Attractor Dynamics approach where spherical coordinates, which represent components of the direction of the movement, model the movement of the robotic manipulator's end-effector. This method was designed to replicate the straight-line motion characteristic of human hand movements. The result consists of a smooth collision-free trajectory continually adjusted according to the target's position and obstacles. An advantage of this approach is that it avoids getting stuck in the local minima by focusing on the trajectory's direction. However, in environments full of obstacles, the attractor dynamics approach faces challenges in ensuring trajectory planarity due to the unevenly repulsive effects distributed across the workspace, leading to weakened avoidance.

2.3.2.3 Dynamic Window Approach (DWA)

Fox et al. (1997) proposed a local motion planning method named the Dynamic Window Approach (DWA) for real-time collision avoidance of mobile robots, which has since been adapted for mobile manipulators (Kobayashi & Motoi, 2022). This sampling-based algorithm is directly informed by the robot's motion dynamics, and it is suitable for robots that operate at high speeds. In this approach, the search space around the robot is reduced to a space of velocities known as the dynamic window. Based on the motion equations of the robot, DWA generates a set of possible circular trajectories composed of translational and angular velocities within the dynamic window, resulting in a two-dimensional velocity search phase. Later, the DWA only considers admissible velocities that allow the robot to stop safely before colliding with any obstacles, removing the velocity profiles that lead to collisions, as the obstacles near the robot impose restrictions on the circular trajectories. The dynamic window further restricts the search space to include only velocities that the robot can reach within a certain time, given the limited accelerations of the robot. Finally, DWA selects the one that maximizes a specific objective function from the remaining velocities. This objective function typically measures the robot's forward velocity, the advance along the path, and the

distance to the nearest obstacle. This approach helps generate stable and smoother paths for the manipulator in static environments compared to other planners like the Time Elastic Band (TEB). However, in highly dynamic environments, sudden changes can lead to manipulator movements that are less feasible.

2.3.2.4 Hybrid Joint/Cartesian DMP-Based Approach

Lauretti et al. (2019) proposed using learning-by-demonstration approaches based on Dynamic Movement Primitives (DMPs) to develop robot movements similar to human behavior. DMPs are a framework of differential equations designed to model and execute robotic movements. The proposed approach is called hybrid joint/Cartesian DMPs (Figure 13), which enables obstacle avoidance and provides anthropomorphism in the cartesian and the joint space. Initially, human motion trajectories are recorded in Cartesian and joint spaces. From the demonstration, distinctive features known as DMP parameters are extracted using a regression algorithm. Subsequently, a dataset of joint and Cartesian DMP parameters is created (offline construction). When the robot is required to perform one of the learned tasks, an online trajectory extraction is made from the set of DMP parameters in the database. Then, the Cartesian and joint DMPs are combined using a coupling equation that prioritizes Cartesian space constraints and calculates the most appropriate reference joint position. Thus, combining these DMPs allows the manipulator to achieve an anthropomorphic configuration in both joint and operational space. For trajectories requiring obstacle avoidance, additional terms are incorporated into Cartesian and joint DMPs that ensure that the end-effector and each manipulator's joint remain free of obstacles within the workspace.

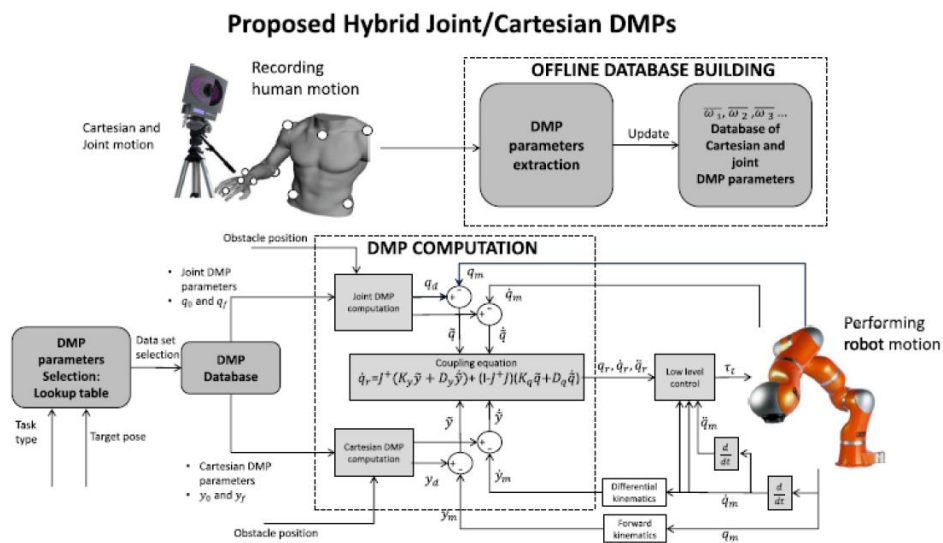


Figure 13 - Block scheme of the Hybrid Joint/Cartesian DMP-Based Approach. Adapted from (Lauretti et al., 2019).

2.4 Motion Planning and Control for Autonomous Mobile Manipulators

The MIAR project aims to control an autonomous mobile manipulator using motion planners similar to those previously analyzed to perform logistics operations such as material handling. The following section presents a study of various case studies that highlight the planning and control of movement for mobile manipulators.

Thakar et al. (2018) proposed a search-based motion-planning approach for mobile manipulators to pick, transport, and place parts between machines and workstations in warehouses and factories. The purpose of the motion algorithm is to generate time-optimal trajectories for the robotic system. To do so, the mobile base's and the manipulator's motion are taken simultaneously, resulting in smaller time operations and more natural movements of the robotic system since the mobile base doesn't stop the manipulator from grasping the items. The mobile base's planning algorithm is based on lattice-based planning for dynamic trajectories (Likhachev & Ferguson, 2009), which uses a discrete search of the configuration space to create the path. For the manipulator, an inverse kinematics solver computes collision-free trajectories, considering the manipulator's joint constraints. The position of the mobile platform is crucial for achieving the fastest path and allowing the robotic arm to reach and grasp the objects effectively.

Annem et al. (2019) presented a teleoperated mobile manipulator system for machine tending tasks and transporting parts within manufacturing environments. The approach results in a combination of semi-autonomy and remote human control. The human operator gives high-level instructions to the mobile manipulator, such as waypoints for the mobile base motion, and sets poses for a 6 DOF manipulator arm equipped with a two-fingered gripper. The mobile base and manipulator comprise multiple sensors to ensure accurate localization on the map and effective obstacle avoidance, respectively. The environment map is made beforehand for the mobile base trajectory generation, and the planning is based on the Dynamic Window Approach (DWA) algorithm. While the mobile base is controlled semi-autonomously, the manipulator remains in its original configuration. The manipulator's

trajectory is generated through the OMPL library, more precisely, the RRT-Connect planner, with the planning framework MoveIt.

Y. Yang et al. (2018) proposed a novel sampling-based motion planning method incorporating a bidirectional search within a time-configuration space. This approach modifies the existing RRT-Connect planner by integrating temporal elements. This was designed for mobile manipulators, or redundant arms, to execute collision-free trajectories for pick-and-place operations of large moving objects in complex and non-static environments. However, prior knowledge of the moving object is required. A practical example was a mobile manipulator that had to pick an object from a moving storage box while navigating.

Engemann et al. (2020) implemented an autonomous omnidirectional mobile manipulator OMNIVIL for flexible logistic operations such as pick, place, and handling. To guarantee efficient human-robot collaboration and precise location of the robotic system during task execution, the mobile manipulator comprises a workspace monitoring system and a set of sensors like 2D and 3D LiDARs and RGB cameras. The control system is divided into lower and higher-level tasks. In the lower level, a PLC controls the servo motors of the mobile base, while in the higher-level control, an industrial computer connects to the PLC and the control unit of the robot arm via ethernet. This system provides autonomous navigation, workspace monitoring, and motion planning of a 6 DOF arm through ROS control. In the workspace monitoring system, the human co-workers are detected, as well as their 2D position, enabling the robot to react preventively to the workspace, reducing its velocity and stopping or changing its planned path. The autonomous navigation is planned using A* or Dijkstra algorithm, while the arm's motion planning is obtained by the RRT-Connect planner using the MoveIt library.

Saoji & Rosell (2020) presented a planning framework for the motion planning of mobile manipulators in dynamic operational environments. In this approach, the Open Motion Planning Library (OMPL) is used for motion planning, while the Fast Forward (FF) task planner is employed for task planning, which uses heuristic search to determine the optimal sequence of actions required to complete a task. Both planning levels are implemented by ROS servers. Given the efficient results, they used the planner RRT-Connect to plan the motion of both the manipulator and the mobile base in the pick-and-place task, even though the planning was conducted separately for each.

Rastegarpanah et al. (2021) proposed a generic framework designed to automate the process of removing and sorting components from electric vehicle batteries with the help of mobile manipulators. The framework enables mobile manipulators to pick up components from a table and place them in the correct bins. The framework comprises navigation, where the robot moves around the workstation, considering the objects in the environment to reach the intended locations. The navigation is handled through a global and local planner approach. The global method uses laser scan sensors that recreate the workspace in a 2D grid map, and the mobile base planning is solved by the A* algorithm. The RRT planner plans the manipulator's trajectory with an inverse kinematics solver to generate a collision-free path. The gripper's grasping actions are controlled via ROS.

3 Theoretical Foundations and Development Tools

This chapter provides the necessary theoretical background to support the concepts applied in this dissertation. Firstly, it aims to clarify some basic concepts of motion planning that are essential for the project's development. While the previous chapter provided an overview of motion planning and included examples of various planners, Section 3.2 highlights the importance of selecting an appropriate motion planner for manipulation tasks and details the planners deployed in the MIAR project. This chapter also introduces the manipulator used in the MIAR project, offering a comprehensive examination of its kinematics properties through the discussion of both forward and inverse kinematics. At last, this chapter offers a comprehensive overview of the software and frameworks utilized in the context of this dissertation.

3.1 Fundamental Concepts of Motion Planning

Within the scope of motion planning, it is important to differentiate two concepts: path planning and trajectory planning. Path planning is a component of the broader motion planning problem aimed at determining a collision-free path between two configurations in the world space without concern for dynamics, the duration of motion, or other constraints (Lynch & Park, 2017). On the other hand, Trajectory planning not only deals with geometric (kinematical) problems but also with dynamic problems, such as managing the system's masses, inertias, actuator limits, and external forces (Choset et al., 2005). It also considers the temporal aspect of motion planning. Trajectory planning uses the path obtained from path planning that indicates the position and orientation of points in cartesian space and joint space and determines the time profiles of position, velocity, and acceleration for each link, which are then provided to the controllers within the robot's control system. The complexity of motion planning increases with the number of degrees of freedom (DOF) in the robot. As the number of paths between the start and goal position increases, so does the number of trajectories, making the problem significantly more challenging and requiring greater computational power. A scheme illustrating how robot motion planning works can be shown in Figure 14.

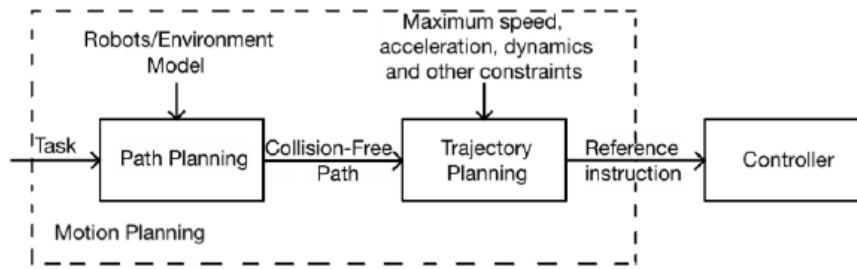


Figure 14 – Schematic representation of robot motion planning. Adapted from (S. Liu & Liu, 2020).

It is important to highlight that the generated planning can occur in two different spaces: Cartesian Space (also known as Task Space or Operational Space) and Joint Space or Configuration Space. The operational space refers to the three-dimensional cartesian space that defines the position and orientation of a robot's end-effector. The joint space is defined by a vector of joint coordinates, whose components are the translational and angular displacements of each joint of a robotic link. Usually, planning movements in Cartesian space requires less computational effort because the trajectories are defined by position and orientation coordinates. In contrast, the joint space has a higher dimensionality, as it depends on each joint of the robot. Since these joints are independent, the search for a motion solution involves a larger search space (Gulletta et al., 2020).

Regarding the operational and joint spaces, the workspace is defined as the subset of the task space that the end-effector's frame can reach. In other words, the workspace of a manipulator is described as a set of points that can be reached by its end-effector and depends on the length of the links and the joint limits. The workspace can further be decomposed into Reachable Space, which consists of the volume of the workspace that the robot can reach from at least one orientation, and dexterous space, which is the volume of the workspace that the robot end-effector can reach in any orientation. The dexterous workspace is a specific part of the reachable workspace. By default, when calculating the workspace of a manipulator, the end-effector is dictated by the end of the last link of the robot. However, depending on the task, the end-effector can vary. For example, in the MIAR Project, the manipulator needs to grasp a set of boxes, so the end-effector, in this case, is the gripper. Thereby, the workspace depends on the tool-frame transformation (Figure 15) that is attached to the tip of the manipulator (Craig, 2017).

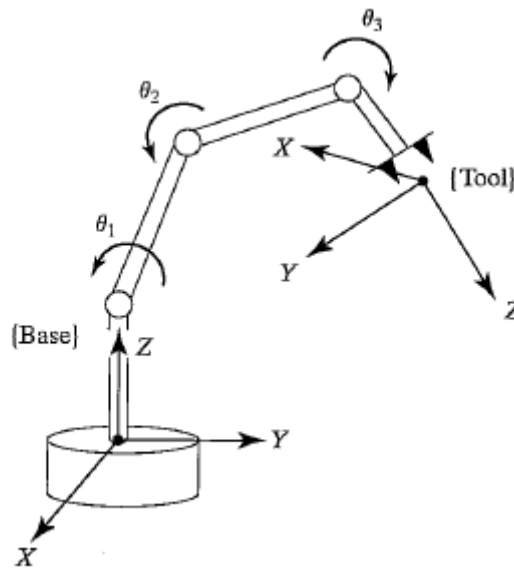


Figure 15 - The representation of a robot's tool frame in relation to the base frame. Adapted from (Craig, 2017).

3.2 Motion Planners Utilized in the MIAR Project

Selecting the most appropriate motion planning algorithm is necessary for a robotic manipulator to execute collision-free paths successfully. However, the selection of the most suitable planner is a difficult assignment since there are numerous effective strategies available for motion planning problems. There isn't a single global planning algorithm suitable for every type of motion planning problem, and all the methods have their weaknesses and strengths (Lynch & Park, 2017). When selecting a motion planning method for robotic systems, several critical factors must be considered to ensure the method is appropriate for a specific application (LaValle, 2006). These factors include: trajectory distance, shorter trajectories often lead to faster task completion; execution time, minimizing cycle time directly impacts productivity in industrial settings; computational complexity, certain algorithms have longer processing times, which can slow down the entire system; planning approach, whether it involves creating a solution based on the robot's environment before the trajectory is initiated or building the trajectory dynamically as the robot moves; robot's dynamic constraints; and the nature of the obstacles, whether the obstacles are rigid or deformable (Gomes De Brito, 2018).

As already stated, the main focus of this dissertation is to develop legible motions for a redundant manipulator in internal logistics tasks, so the selection of the ideal motion planner

is a topic of crucial importance in the development of this work. The manipulator has to operate in a restricted space where the placement of the boxes in the access ramps presents a significant challenge. As a result, it is required to create a full trajectory representing all the operations of pick or place, considering all the obstacles within the manipulator's workspace that could affect its planning. Given the nature of these environments, a global motion planner is used to execute the intended tasks.

Although the collaborative manipulator will operate in an environment shared with human operators, it will not perform collaboratively directly with them. In other words, the manipulator will not directly interact with the agents around it. Hence, this project excludes the use of human-like planners, as the primary goal is to generate safe and smooth trajectories for the manipulator while ensuring it can complete tasks in the shortest time possible, which is critical to meet work deadlines. Human-like planners are typically slower and demand greater computational resources, making them less suitable for this application.

The global planners used in this project are the RRT, RRT-Connect, RRT*, T-RRT, Bi-TRRT, PRM*, LazyPRM*, ProjEST, BiEST, and KPIECE. The planners will be presented in more detail next, except for the RRT, RRT-Connect, and RRT* planners, which have already been analyzed in Chapter 2.3.

3.2.1 PRM*

PRM* is an advanced version of the standard sampling-based motion planning algorithm PRM designed to find optimal paths in high-dimensional configuration spaces. This extended method approaches optimality by systematically increasing the number of connection attempts as the roadmap expands. As already explained, the Probabilistic Roadmap strategies generate nodes through random samples of the robot's configuration space. While PRM connects each node to its nearest neighbors from a fixed radius, PRM* uses a sphere with a connection radius for each newly sampled node. The radius of these spheres is scaled logarithmically based on the number of neighboring nodes around the new node (Karaman & Frazzoli, 2011). This method ensures that every node is linked to enough neighbors, facilitating the discovery of shorter paths. Therefore, this approach makes the roadmap by connecting the nodes within the specified radius. The algorithm uses graph search techniques, like Dijkstra's or A* algorithms, to find the shortest path between the start and goal configurations within the roadmap. In this planner, the phases of building the roadmap

and searching occur simultaneously, and like the RRT* planner, PRM* is also considered asymptotic optimality (Alandes, 2015).

3.2.2 LazyPRM*

The LazyPRM* is an asymptotically optimal sampling-based motion planning algorithm that uses lazy collision checking to enhance pathfinding efficiency in high-dimensional configuration spaces, meaning that it only defers collision checks on an edge when it is part of a potential candidate path. As the name suggests, this is a version of PRM* (Hauser, 2015). While PRM* performs collision checks during the roadmap construction phase, leading to higher computational costs, Lazy PRM integrates lazy collision checking. Still, it does not guarantee optimality, LazyPRM* incorporates lazy state validity checking into the PRM* planner. This algorithm creates random nodes, and each node is connected to its neighbors based on a connection radius to discover optimal paths. Unlike PRM*, the edges between the nodes aren't immediately checked for collisions. Edges are assumed to be initially valid and are checked only when a candidate's path to the goal is found. If an edge is detected to be in collision in the checking, it is removed from the roadmap, and alternative paths are explored. This strategy avoids many edges that cannot be on an optimal path, significantly reducing computational overhead.

3.2.3 KPIECE

The Kinodynamic Planning by Interior-Exterior Cell Exploration (KPIECE) algorithm (I. Sutan & Kavraki, 2008) is a sampling-based motion planner specifically designed to handle systems with complex dynamics. Unlike the vast majority of planners, which require explicit state sampling, KPIECE projects the high-dimensional state space onto a one or multi-level grid-based discretization, creating a set of cells that represent different regions of the space. This discretization guides the planner in estimating the coverage of the state space and allows for identifying underexplored areas. Since KPIECE is a tree-based planner, it creates a tree of motion from the initial state that grows through the cells, connecting feasible new states. As new states are added, KPIECE updates the coverage information for the corresponding cells and adjusts its exploration strategy to concentrate on areas of the grid with less coverage. During the exploration, the algorithm distinguishes between interior cells, surrounded by explored cells, and exterior cells, which are on the boundary of the explored regions. By taking

preference on exterior cells, KPIECE expands the boundaries of the explored space, promoting a complete exploration. The process continues until a path from the initial state to the goal region is found.

3.2.4 T-RRT

The Transition-based Rapidly Exploring Random Tree (T-RRT) is a sampling-based motion planning algorithm that computes paths in high-dimensional cost spaces by combining randomized path planning and stochastic optimization methods (Jaillet et al., 2010). This is an extended version of the RRT algorithm, which utilizes the exploratory strengths of the planner to navigate through the space. T-RRT incorporates a stochastic transition test to guide the exploration toward lower-cost areas within the robot's configuration space. T-RRT uses this transition test to decide whether to accept or reject new potential states to the growing tree based on the cost function variation relative to the local motion that connects the current state to the new potential state (Figure 16). The cost function considered in the costmap can represent various criteria, such as energy consumption or the distance to maximize between the robot and surrounding objects to create paths with high clearance. Therefore, T-RRT favors solutions that minimize the accumulated cost, leading to paths that are considered not only feasible but also safe. The planner also has a temperature parameter that is dynamically adjusted through the search process and influences the acceptance probability of states. At each accepted uphill transition, this parameter is decreased to avoid over-exploring in regions of high cost, and in case the uphill transition between the states is rejected, the temperature is increased to help the exploration to find new potential states (Devaurs et al., 2013). Thus, this algorithm starts by initializing a tree with an initial configuration and setting the initial temperature parameter. The algorithm samples a random state, and the existing tree attempts to extend toward the sampled state, generating a potential new one. The cost of the new state is compared to the cost of the parent state, and the transition test determines the acceptance probability based on the cost difference. If the new state is accepted, it is added to the tree. At the same time, the temperature parameter is adjusted based on the success rate of the state transition, allowing the algorithm to adapt dynamically to the cost map. This process repeats until the tree reaches the goal state. Compared to the RRT planner, the T-RRT produces better paths because the RRT explores the configuration space to find feasible paths but doesn't consider a cost function. On the other hand, T-RRT isn't asymptotically optimal,

meaning it doesn't converge towards an optimal solution like the RRT* planner, which aims to find the shortest path in configuration space by continuously refining its tree structure. Nevertheless, T-RRT offers faster and more feasible paths in high-dimensional spaces than RRT*, which can take time to find a solution.

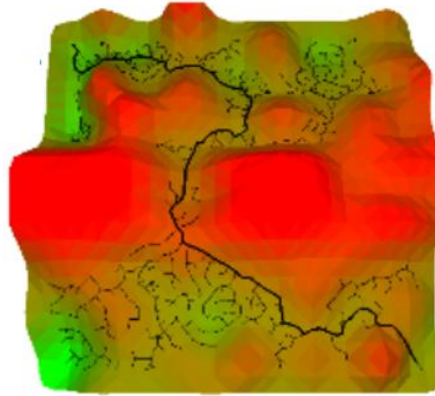


Figure 16 - Example of a construction process of the Transition-based RRT planner. The red zones represent high cost area of the cost map. Adapted from (Jaillet et al., 2010).

3.2.5 Bi-TRRT

The Bidirectional Transition-based Rapidly Exploring Random Tree (Bi-TRRT) is an adaption of the T-RRT algorithm that merges the advantages of bidirectional search with cost-sensitive exploration (Devaurs et al., 2013). Similar to the RRT-Connect planner, Bi-TRRT considers two growing trees, one rooted from the initial configuration and the other from the goal state, and tries to connect them. Each tree explores the cost map independently and has its own temperature parameter. The algorithm alternates between growing each tree. At each iteration, one of the trees attempts to extend toward a sampled random state of the configuration space. Then, the transition test determines whether the new state is accepted or rejected based on the cost difference between the new and the parent state. If the new state passes the transition test, it is added to the tree, and the other tree tries to expand toward this new node. If the trees are unable to connect, they switch roles, and the process is repeated. This process continues with the trees alternately expanding toward randomly sampled states until the junction is made. While RRT* generally converges to better solutions compared to Bi-TRRT, the latter often discovers solutions more quickly.

3.2.6 BiEST

Expansive Space Trees (EST) is a tree-based motion planning algorithm built for robots with many DoFs that incrementally builds a roadmap through the configuration space, favoring the exploration of less explored areas through the concept of visibility (Hsu et al., 1997). The planner comprises the expansion and connection phases, which are iteratively executed. In the expansion phase, a node in the tree is selected for expansion based on its weight function, which is defined by the number of sampled configurations within its neighborhood up to a specified distance threshold. Nodes with lower weights, indicating fewer nearby configurations, are prioritized for expansion. This ensures the tree grows toward less explored areas, preventing repeated exploration of already covered regions and encouraging expansion into new directions. Then, random states are sampled in the neighborhood of the selected node of the tree. The group of nodes near the free node that can connect in a straight line without encountering obstacles is called the visibility set (Alandes, 2015). The planner connects those sampled configurations that contribute to the visibility region with the free node, leading to the expansion of the tree. Then, the process repeats. In the connection phase, the algorithm tries to connect the tree to the goal configuration. If the connection is not possible, the tree continues to expand. While the PRM planner generates random nodes throughout the configuration space and tries to connect them, later searching for the best path between the initial and goal nodes, the EST planner focuses its exploration solely on the relevant space regions to reach the solution. This approach allows the EST to avoid unnecessary computation when creating a roadmap for the entire configuration space. Therefore, this planner EST is ideal for single-query path planning problems and is well suited for environments with narrow passages.

The BiEST algorithm is a bidirectional version of the EST planner that explores the configuration space by growing two trees simultaneously, one from the start configuration and another from the goal configuration. For each tree, it expands toward the node with the lowest weight, and random configurations are sampled. The tree expands toward the configurations within the visibility region of the selected node. The two trees grow toward each other, and when the visibility region of one of the trees intersects with the other, the planner attempts to connect the newly added nodes. The visibility region of a tree is the union of the visibility regions of all its nodes. As the two trees grow toward each other, the

exploration space decreases, allowing the BiEST planner to find solutions more quickly than the EST planner.

3.2.7 ProjEST

The Projection-based Expansive Space Tree algorithm (ProjEST) is an extended version of the EST planner that incorporates a projection function to map the high-dimensional state space into a lower-dimensional space (Hsu et al., 1997). This projection enables ProjEST to impose a grid on the projected space, allowing it to focus the expansion of the tree in underexplored areas. The algorithm selects nodes for expansion based on a weight function, prioritizing regions that are less densely sampled, and uses the projection to guide the sampling and connection of the nodes near the tree. The planner uses a default projection associated with the state space if no projection is defined. The algorithm attempts to connect the tree, which roots from the initial node, to the goal configuration, iterating until a valid path is discovered. By reducing the dimensionality of the problem during exploration, ProjEST can converge on solutions more quickly than the standard EST planner.

3.3 Hardware Components

The robotic arm defined for the motion planning tasks in the scope of the MIAR project is the collaborative manipulator KUKA LBR iiwa 14 R820 (Figure 17). This is a redundant manipulator with 7 degrees of freedom. A manipulator is considered kinematically redundant when it has more degrees of freedom than those strictly necessary to perform a given task. Manipulators with 7 DoF are generally redundant, which allows for greater manipulability. The reason for using this type of robot in the case study is primarily due to its redundancy, which enables more precise and dexterous movements. This is beneficial for performing tasks and allowing the robotic arm to exhibit smoother and legible movements. The redundant degree of freedom allows the manipulator to perform secondary tasks beyond achieving a specific position and orientation, such as avoiding obstacles. The 7-DoF anthropomorphic robot features an S-R-S kinematic structure consisting of a spherical shoulder, a rotational elbow, and a spherical wrist (Faria et al., 2018).

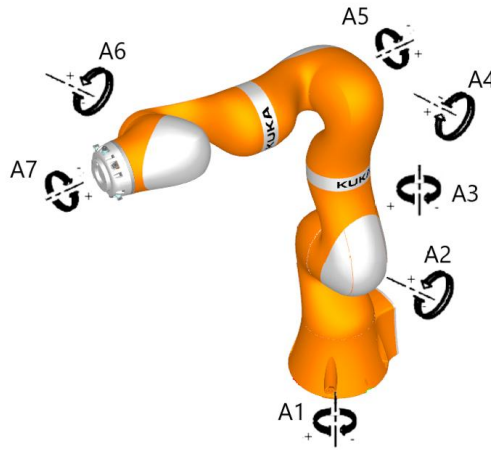


Figure 17 - Joint angles of the KUKA LBR iiwa 14 R820.

This manipulator has a maximum reach of 820 mm, a rated payload of 14 kg, a pose repeatability of ± 0.15 mm, and a weight of approximately 29.9 kg. It is designed to operate within an ambient temperature range of -5°C to 45°C . Additional characteristics, such as axis data, are provided in Table 1.

Table 1 - Motion range and speed specifications of the KUKA LBR iiwa 14 R820 joints.

	Joints						
	A1	A2	A3	A4	A5	A6	A7
Motion range	± 170	± 120	± 170	± 120	± 170	± 120	± 175
Speed with rated payload	85 °/s	85 °/s	100 °/s	75 °/s	130 °/s	135 °/s	135 °/s

The graphical representation of the manipulator's workspace can be found in Annex A.

The kinematic analysis of a robotic arm is essential for understanding and controlling its motion, enabling it to execute specific operations. Studying the kinematics model of robotic arms with 7 DoF can be particularly challenging. This subchapter presents the theoretical foundations for solving the Forward and Inverse Kinematics of the project's manipulator.

3.3.1 Forward Kinematics

The forward kinematics (FK) involves determining the position and orientation of a robotic arm's end-effector based on the robot's joint configuration. In other words, by using

the values from the robot's joints, the robot's pose is determined, thus converting from the joint space to the cartesian space. The FK of manipulators usually uses the Denavit-Hartenberg classic convention (Craig, 2017). Firstly, in this approach, the manipulator is considered at a given position, typically the “zero” position of the robot, where all joint values are zero. Then, a coordinate frame is attached to each robot link, from the base to the tip, enumerated from 0 to n. The coordinate axes assigned to each link are set at the furthest point of the link. The z-axis aligns with the joint axis, and its direction is established by the positive direction of rotation of the joint if it is a rotational joint or along the motion direction of the link if it is a prismatic joint. The x-axis is perpendicular to the z-axis, while the right-hand rule gives the direction of the y-axis. Therefore, the representation of the axes on the robot KUKA LBR iiwa 14 R820 is depicted in Figure 18.

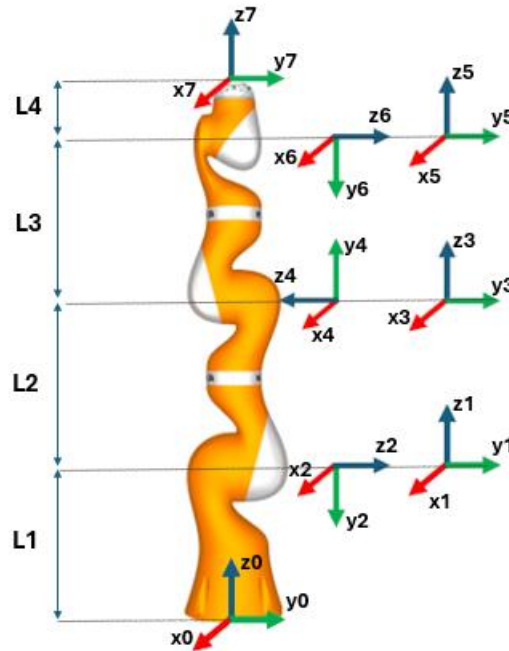


Figure 18 - Assignment of the D-H coordinate frames to each joint of the KUKA LBR iiwa 14 R820 robot.

Subsequently, the D-H parameters are used to describe the position and orientation of each link's frame in relation to the frame of the previous link. These parameters are categorized into two types: variables and constants. The variables include θ_n , which represents the angular rotation relative to the axes z_{n-1} and z_n , and d_n , which indicates the displacement from frame n-1 to frame n. The constants are α_n , corresponding to the angle concerning the axes x_{n-1} and x_n , and l_n , which denotes the distance from frame n-1 to frame

n. Based on the coordinate system shown above, the D-H table of the manipulator under study is illustrated in Table 2.

Table 2 - Denavit-Hartenberg parameters table for the KUKA LBR iiwa 14 R820 robot.

Link n	Variable	θ_n	α_n	d_n	l_n
1	θ_1	θ_1	$-\frac{\pi}{2}$	L1	0
2	θ_2	θ_2	$\frac{\pi}{2}$	0	0
3	θ_3	θ_3	$\frac{\pi}{2}$	L2	0
4	θ_4	θ_4	$-\frac{\pi}{2}$	0	0
5	θ_5	θ_5	$-\frac{\pi}{2}$	L3	0
6	θ_6	θ_6	$\frac{\pi}{2}$	0	0
7	θ_7	θ_7	0	L4	0

For each row in the table, the homogeneous transformation matrix ${}^{n-1}T_n$ is obtained, indicating the frame's position and orientation with respect to the prior frame. The matrix is calculated by multiplying the transformations of the D-H parameters, prioritizing rotation followed by translation, with the z-axis taking priority over the x-axis, as can be seen in Equation 3.1.

$${}^{n-1}T_n = \text{Rot}(Z_{n-1}, \theta) \text{Trans}(0,0,d) \text{Trans}(l,0,0) \text{Rot}(x_n, \alpha) =$$

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \cos(\alpha) & \sin(\theta) \sin(\alpha) & l \cos(\theta) \\ \sin(\theta) & \cos(\theta) \cos(\alpha) & -\cos(\theta) \sin(\alpha) & l \sin(\theta) \\ 0 & \sin(\alpha) & \cos(\alpha) & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad 3.1$$

In turn, the matrices are “chained together” (Alberto & Borges, 2022) through the product of the homogenous matrices to create the general transformation matrix, which indicates the pose of the robot relative to the base (Equation 3.2).

$${}^0T_7 = {}^0T_1 {}^1T_2 {}^2T_3 {}^3T_4 {}^4T_5 {}^5T_6 {}^6T_7 \quad 3.2$$

The individual homogenous matrices of the robot KUKA LBR iiwa 14 R820 can be checked in Equation 3.3, where L1 (link 1) is the distance from the robot’s base to its shoulder, L2 (link 2) is the distance from the shoulder to the elbow, L3 (link 3) is the distance from elbow to wrist and L4 (link 4) is the distance from wrist to tip/end-effector, respectively (Figure 18).

$$\begin{aligned} {}^0T_1 &= \begin{bmatrix} \cos(\theta_1) & 0 & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & 0 & \cos(\theta_1) & 0 \\ 0 & -1 & 0 & L1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ {}^1T_2 &= \begin{bmatrix} \cos(\theta_2) & 0 & \sin(\theta_2) & 0 \\ \sin(\theta_2) & 0 & -\cos(\theta_2) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ {}^2T_3 &= \begin{bmatrix} \cos(\theta_3) & 0 & \sin(\theta_3) & 0 \\ \sin(\theta_3) & 0 & -\cos(\theta_3) & 0 \\ 0 & 1 & 0 & L2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ {}^3T_4 &= \begin{bmatrix} \cos(\theta_4) & 0 & -\sin(\theta_4) & 0 \\ \sin(\theta_4) & 0 & \cos(\theta_4) & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ {}^4T_5 &= \begin{bmatrix} \cos(\theta_5) & 0 & -\sin(\theta_5) & 0 \\ \sin(\theta_5) & 0 & \cos(\theta_5) & 0 \\ 0 & -1 & 0 & L3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ {}^5T_6 &= \begin{bmatrix} \cos(\theta_6) & 0 & \sin(\theta_6) & 0 \\ \sin(\theta_6) & 0 & -\cos(\theta_6) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ {}^6T_7 &= \begin{bmatrix} \cos(\theta_7) & -\sin(\theta_7) & 0 & 0 \\ \sin(\theta_7) & \cos(\theta_7) & 0 & 0 \\ 0 & 0 & 1 & L4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad 3.3$$

The position of the end-effector is represented by the first three rows of the fourth column of the matrix ${}^0T_7$ derived from the Equation 3.2. Its orientation is determined by the rotation transformation matrix, which aligns with the matrix's first three rows and columns. The end-effector's orientation relates to the joint movements of the robot's wrist. This orientation is described by the accumulation of three rotations around the z, y, and x axes in relation to the base frame, known as Roll, Pitch, and Yaw. Specifically, Yaw (Y) refers to the rotation around the x-axis, Pitch (β) corresponds to the rotation around the y-axis, and Roll (α) pertains to the rotation around the z-axis. According to the Euler convention, the rotation matrix through the general orientation transformation of the Roll-Pitch-Yaw type is obtained by Equation 3.4.

$$R_{XYZ}(Y, \beta, \alpha) = R(Z, \alpha) R(Y, \beta) R(X, Y) =$$

$$\begin{bmatrix} \cos(\alpha) \cos(\beta) & \cos(\alpha) \sin(\beta) \sin(Y) - \sin(\alpha) \cos(Y) & \cos(\alpha) \sin(\beta) \cos(Y) + \sin(\alpha) \sin(Y) \\ \sin(\alpha) \cos(\beta) & \sin(\alpha) \sin(\beta) \sin(Y) + \cos(\alpha) \cos(Y) & \sin(\alpha) \sin(\beta) \cos(Y) - \cos(\alpha) \sin(Y) \\ -\sin(\beta) & \cos(\beta) \sin(Y) & \cos(\beta) \cos(Y) \end{bmatrix} \quad 3.4$$

Therefore, the three angles can be computed by the following Equations 3.5, 3.6 and 3.7.

$$\alpha = \text{atan2}(r_{21}, r_{11}) \quad 3.5$$

$$\beta = \text{atan2}(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}) \quad 3.6$$

$$Y = \text{atan2}(r_{32}, r_{33}) \quad 3.7$$

If the angle $\beta = \pm 90^\circ$, the solution degenerates. A possible convention is to consider the α angle equal to 0, as presented in Equations 3.8 and 3.9.

$$\beta = 90^\circ \rightarrow \begin{cases} \alpha = 0 \\ Y = \text{atan2}(r_{12}, r_{22}) \end{cases} \quad 3.8$$

$$\beta = -90^\circ \rightarrow \begin{cases} \alpha = 0 \\ Y = -\text{atan2}(r_{12}, r_{22}) \end{cases} \quad 3.9$$

In this way, the pose of the robot's end-effector is obtained (Equation 3.10).

$${}^0p_7 = (xe, ye, ze, Y, \beta, \alpha) \quad 3.10$$

Since this robot has 7 DoF, it is considered a redundant robot. Robots with a redundant degree of freedom exhibit greater manipulability, being able to execute operations while taking into consideration obstacles, joint limits, and singularities avoidance (Faria et al., 2018). For a given pose of the robot's tip, the elbow can rotate along a circular arc whose normal vector is aligned with the axis running from the shoulder to the wrist. Hence, the concept of the redundancy circle emerged, referring to the redundant motion space of the elbow in these types of manipulators. Determining the elbow position is essential for solving the FK problem. To achieve this, the parameter known as the arm angle (Ψ), presented by Hollerbach (1985) specifies the elbow position in the redundancy circle. The arm angle is the angle spawned by the plane of the real arm's shoulder, elbow, and wrist positions (SEW plane) and a virtual plane of these joints in a non-redundant image of the manipulator by restricting the value of the joint 3 to zero (SEWv), that serves as the reference plane (Shimizu et al., 2008). The illustration of the arm angle is depicted in Figure 19. To obtain the elbow position, it is essential to know the values of the virtual joints θ_{1v} , θ_{2v} , θ_{3v} and θ_{4v} , once these are the angles that guarantee that the virtual manipulator has the same tip pose as the real manipulator. As mentioned earlier, in the virtual manipulator θ_{3v} is 0 and θ_{4v} is equal to θ_4 because the positions of the shoulder and wrist in the virtual and real manipulator are the same (Alberto & Borges, 2022). As a result, θ_{1v} and θ_{2v} are the only virtual angles that need to be determined in the FK.

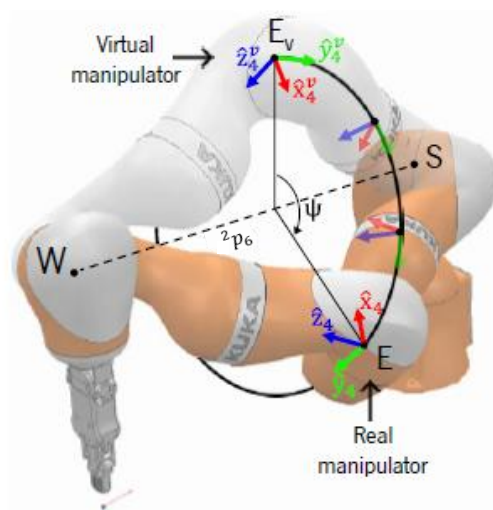


Figure 19 - Representation of the arm angle parameter in the redundancy circle. Adapted from (Alberto & Borges, 2022).

According to Figure 17, the angle θ_{1v} is located on the xy plane since the robot's first joint rotates around the horizontal plane. To obtain the angle, it is necessary to determine the distance from the robot's shoulder to its wrist. As the position and orientation of the robot's tip are known due to the general homogenous transformation matrix and the vector from the base to the shoulder is ${}^0p_2 = [0 \ 0 \ L1]$, because the link 1 only rotates axially, then the vector from the shoulder to the tip is computed by the Equation 3.11.

$${}^2p_7 = {}^0p_7 - {}^0p_2 \quad 3.11$$

Since the vector from the wrist to the robot's tip is ${}^6p_7 = [0 \ 0 \ L4]$ and the real orientation between these two frames can be acquired by the rotation matrix ${}^0R_7$ derived from ${}^0T_7$, as presented above in Equation 3.2, the shoulder-wrist vector is calculated using the Equation 3.12.

$${}^2p_6 = {}^2p_7 - {}^6p_7 {}^0R_7 \quad 3.12$$

The θ_1 angle, positioned in the xy-plane, establishes the x and y coordinates of the tip and, consequently those of the wrist, and is thus influenced by ${}^2p_{6,x}$ and ${}^2p_{6,y}$. However, in case the vector ${}^2p_6$ is colinear with the standard unit vector in the direction of the x-axis $[0,0,1]$, which means that the wrist of the robot is aligned with the shoulder direction, then θ_{1v} is undefined because ${}^2p_{6,x}$ and ${}^2p_{6,y}$ are zero. This scenario represents a singularity. To address this issue, when such a condition arises, θ_{1v} takes the value 0. This condition is verified by the cross product of both colinear vectors, which must be 0 (Equation 3.13).

$$\theta_{1v} = \begin{cases} \text{atan2}({}^2p_{6,y}, {}^2p_{6,x}), & \text{if } \|{}^2p_6 \times [0 \ 0 \ 1]\| > 0 \\ 0, & \text{if } \|{}^2p_6 \times [0 \ 0 \ 1]\| = 0 \end{cases} \quad 3.13$$

As shown in Figure 20, the virtual joint θ_{2v} is based on the projection of the shoulder-wrist vector on the z-axis and the xy-plane, but it also depends on the angle Φ , the latter being determined by the law of cosines (Equation 3.14).

$$\Phi = \arccos\left(\frac{d_{se}^2 + \|{}^2p_6\|^2 - d_{ew}^2}{2 d_{se} \|{}^2p_6\|}\right) \quad 3.14$$

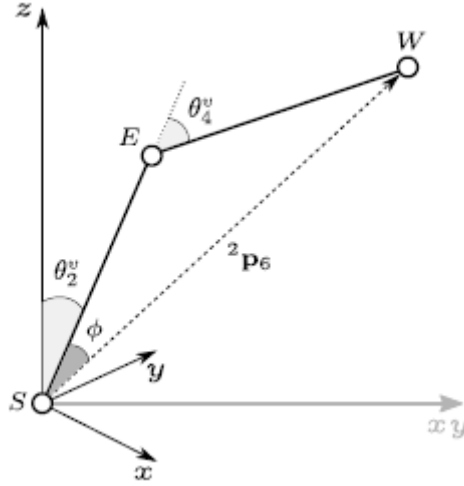


Figure 20 - Illustration of the virtual manipulator used to calculate FK. Adapted from (Faria et al., 2018).

Depending on whether the elbow of the robot is upwards or downwards, the value of θ_{2v} will differ. That being said, another parameter, called Global Configuration (GC), is considered. This parameter is associated with specific joints and takes a value of 1 if the joint's configuration is positive and -1 otherwise. In this case, if the elbow is up ($\theta_4 > 0$), the GC will be 1, and if the elbow is down, the GC will be -1 (Equation 3.15).

$$\theta_{2v} = \text{atan2}\left(\sqrt{{}^2p_{6,x}^2 + {}^2p_{6,y}^2}, {}^2p_{6,z}\right) + GC_4\Phi \quad 3.15$$

With the virtual angles known, the virtual position of the robot's shoulder (${}^0p_{4v}$) can be determined by applying the previously described FK process to obtain the homogeneous transformation matrix ${}^0T_{4v}$. Given that, the shoulder and wrist positions are identical in both the real and virtual configurations, as seen in Figure 19, the vectors linking the base to the shoulder (${}^0p_2$ and ${}^0p_{2v}$) and the base to the wrist (${}^0p_6$ and ${}^0p_{6v}$) are therefore equivalent. The only difference lies in the vectors linking the base to the elbow (${}^0p_4$ and ${}^0p_{4v}$) owing to the different elbow positions in the real and virtual manipulators. As the arm angle is defined as

the angle between the two planes: SEW and SEWv, the parameter will be computed by calculating the angle between the normal vectors of these planes.

The cross-product of two vectors spawns a vector that is orthogonal to both. The normal vector of the plane SEW can be determined by the cross-product of the vectors belonging to the plane, namely the vectors linking the shoulder to the elbow (${}^2p_4$) and the elbow to the wrist (${}^2p_6$) (Equation 3.18).

$${}^2p_4 = {}^0p_4 - {}^0p_2 \quad 3.16$$

$${}^2p_6 = {}^0p_6 - {}^0p_2 \quad 3.17$$

$$V_{SEW} = {}^2p_4 \times {}^2p_6 \quad 3.18$$

The normal vector for the plane SEWv is through the vectors ${}^2p_{4v}$ and ${}^2p_{6v}$ (Equation 3.21).

$${}^2p_{4v} = {}^0p_{4v} - {}^0p_{2v} \quad 3.19$$

$${}^2p_{6v} = {}^0p_{6v} - {}^0p_{2v} \quad 3.20$$

$$V_{SEWv} = {}^2p_{4v} \times {}^2p_{6v} \quad 3.21$$

Before formulating the arm angle, it is important to find whether the parameter is positive or negative. The cross product of the normal vectors defined in Equations 3.18 and 3.21 generates a vector that is collinear with the vector that connects the shoulder to the wrist (${}^2p_6$). If these vectors have the same direction, the sign of the arm angle is positive, otherwise, it is negative. To conclude, the arm angle results from the dot product to the two normal vectors VSEW and VSEWv (Equation 3.22).

$$\Psi = \text{sign}_{\Psi} \arccos(\widehat{V_{SEWv}} \cdot \widehat{V_{SEW}}) \quad 3.22$$

3.3.2 Inverse Kinematics

The inverse kinematics (IK) consists of finding the joint values of the robot based on the position and orientation coordinates of the end-effector, that is, there is a conversion from the cartesian space to the joint space. As previously indicated, 7-DOF manipulators are generally redundant, and this redundancy problem is addressed in the IK. As the target pose is specified by six variables (three for position and three for orientation), it becomes necessary to solve a system of six nonlinear equations involving seven unique variables corresponding to the robotic arm's joints. This results in an indeterminate system, indicating that the inverse kinematics problem is under-constrained. Faria et al. (2018) proposed an analytical approach for solving the inverse kinematics of 7-DOF manipulators designed to prevent joint limits and singularities. For this purpose, beyond the coordinates of the desired pose, the parameters arm angle (Ψ) and global configuration (GC) are introduced as inputs to the problem. Thus, solving the arm's inverse kinematics requires these three variables, whereby for a given target pose, the desired arm angle is specified, along with the GC values for the shoulder, elbow, and wrist joints (values of 1 or -1), indicating whether θ_2 , θ_4 , and θ_6 are positive or negative. In IK for 7-DOF manipulators, there are eight possible solutions for a specific hand pose and known arm angle: four with the elbow (θ_4) bent backward and another four with the elbow bent forward. Within each of these eight configurations, there is an infinity of possibilities in the null space due to the seventh degree of freedom, which allows continuous movements of the robot without affecting the end-effector's pose.

According to Faria et al. (2018), to determine the real joints of the robot, it is necessary to first calculate the value of joint θ_4 , as it is the only joint independent of the arm angle parameter. As such, by applying the Equation 3.12 again, it's possible to have the real shoulder-wrist vector. Since the shoulder position doesn't change relative to the base, the vector ${}^0p_2$ remains the same as the virtual one, the vector ${}^0p_7$ is derived from the desired position, the matrix ${}^0R_7$ is established using the desired orientation of the robot's end-effector (Yaw, Pitch, Roll) and the Roll-Pitch-Yaw transformation matrix (Equation 3.4), and the vector ${}^6p_7$ is once again the vector that links the wrist to the end-effector by the z-axis.

Thus, based on Figure 20, the angle θ_4 is achievable through the law of cosines. As mentioned earlier, in the IK, the GC is an input that describes the robot's configuration. Since the remaining joint angles depend on the upward or downward orientation of the robot's

elbow, it is essential to introduce the GC4 parameter in Equation 3.23 to specify whether θ_4 will be positive or negative.

$$\theta_4 = GC4 \arccos \left(\frac{\|{}^2p_6\|^2 - d_{se}^2 - d_{ew}^2}{2 d_{se} d_{ew}} \right) \quad 3.23$$

In order to discover the other joint angles, the real manipulator's elbow position must be found. To accomplish this, it is first calculated the pose of the virtual elbow, using the process previously applied in the FK, whereby θ_{1v} , Φ angle and θ_{2v} are determined with the help of Equations 3.13, 3.14 and 3.15. It is important to remind that the value of the virtual joint θ_{3v} is zero. The pose of the virtual elbow is dictated by the ${}^0T_{3v}$ or ${}^0T_{4v}$ matrices through the D-H convention, where ${}^0R_{3v}$ or ${}^0R_{4v}$ represent the orientation of the frames in the elbow presented in the Figure 18.

Considering that the arm angle defines the angle between the real arm configuration and its virtual/non-redundant configuration, the real elbow pose will correspond to the virtual elbow pose when rotated by an Ψ angle around the axis passing through the elbow and wrist (Equation 3.24).

$${}^0R_3 = {}^0R_\Psi {}^0R_{3v} \quad 3.24$$

Given this logic, Rodrigues's rotation formula can be used to convert the virtual elbow pose to the real elbow pose. The Rodrigues's rotation formula allows the rotation of the vectors from the rotation matrix around a specified axis by a given angle (Equation 3.25).

$$R = I_3 + (\sin \varphi) K + (1 - \cos(\varphi)) K^2 \quad 3.25$$

, where I_3 is the identity matrix, φ is the rotation angle around the axis, and K is the cross-product matrix associated with the unit vector k , that represents the rotation axis. Applying this formula to the case study results in the following Equation 3.26.

$${}^0R_\Psi = I_3 + (\sin \Psi) [\widehat{{}^2p_6} \times] + (1 - \cos(\Psi)) [\widehat{{}^2p_6} \times]^2 \quad 3.26$$

Replacing the Equation 3.26 on the Equation 3.24, ${}^0R_3$ can be formulated as follows:

$${}^0R_3 = A_s \sin(\Psi) + B_s \cos(\Psi) + C_s \quad 3.27$$

, where:

$$\begin{aligned} A_s &= [\widehat{{}^2p_6} \times] {}^0R_{3v} \\ B_s &= -[\widehat{{}^2p_6} \times]^2 {}^0R_{3v} \\ C_s &= [\widehat{{}^2p_6} \widehat{{}^2p_6}^T] {}^0R_{3v} \end{aligned}$$

The matrix ${}^0R_3$ can also be expressed using the Denavit-Hartenberg convention (Equation 3.28) through the multiplication of ${}^0T_1$, ${}^1T_2$ and ${}^2T_3$, as previously detailed.

$$\begin{aligned} {}^0R_3 &= \begin{bmatrix} \cos(\theta_1) \cos(\theta_2) \cos(\theta_3) - \sin(\theta_1) \sin(\theta_3) & \cos(\theta_1) \sin(\theta_2) & \cos(\theta_3) \sin(\theta_1) + \cos(\theta_1) \cos(\theta_2) \sin(\theta_3) \\ \cos(\theta_1) \sin(\theta_3) + \cos(\theta_2) \cos(\theta_3) \sin(\theta_1) & \sin(\theta_1) \sin(\theta_2) & \cos(\theta_2) \sin(\theta_1) \sin(\theta_3) - \cos(\theta_1) \cos(\theta_3) \\ -\sin(\theta_2) \cos(\theta_3) & \cos(\theta_2) & -\sin(\theta_2) \sin(\theta_3) \end{bmatrix} \quad 3.28 \end{aligned}$$

By analyzing the matrix ${}^0R_3$, it can be observed that θ_2 can be determined as follows:

$$\theta_2 = \arccos(a_{s32} \sin(\Psi) + b_{s32} \cos(\Psi) + c_{s32}) \quad 3.29$$

Based on Equation 3.29, and taking into consideration the signal of θ_2 given by the global configuration parameter (GC2), it can be concluded that the real joint θ_2 can be expressed as follows:

$$\theta_2 = \text{GC2} \arccos(a_{s32} \sin(\Psi) + b_{s32} \cos(\Psi) + c_{s32}) \quad 3.30$$

Applying the same logic, it is possible to determine the values of the joints θ_1 and θ_3 , which depend on θ_2 .

$$\theta_1 = \text{atan2}(\text{GC2} [a_{s22} \sin(\Psi) + b_{s22} \cos(\Psi) + c_{s22}], \text{GC2} [a_{s12} \sin(\Psi) + b_{s12} \cos(\Psi) + c_{s12}]) \quad 3.31$$

$$\theta_3 = \text{atan2}(\text{GC2} [-a_{s33} \sin(\Psi) - b_{s33} \cos(\Psi) - c_{s33}], \text{GC2} [-a_{s31} \sin(\Psi) - b_{s31} \cos(\Psi) - c_{s31}]) \quad 3.32$$

The calculation of the remaining joints of the robot will depend on ${}^0R_3$, which can now be determined since the joints θ_1 , θ_2 , and θ_3 are known; on ${}^0R_7$, which is defined by the robot's pose, as established in Equations 3.4 and 3.10; and on ${}^3R_4$, which is obtained from the matrix ${}^3T_4$, presented in Equation 3.3, and the value of joint θ_4 , which is now known. With all these matrices recognized, the matrix ${}^4R_7$ can be expressed by Equation 3.33.

$${}^4R_7 = {}^3R_4^T {}^0R_3^T {}^0R_7 \quad 3.33$$

Following the same reasoning presented earlier for the representation of the ${}^0R_3$ matrix, ${}^4R_7$ can be expressed as follows:

$${}^4R_7 = A_w \sin(\Psi) + B_w \cos(\Psi) + C_w \quad 3.34$$

, where:

$$\begin{aligned} A_w &= {}^3R_4^T A_s^T {}^0R_7 \\ B_w &= {}^3R_4^T B_s^T {}^0R_7 \\ C_w &= {}^3R_4^T C_s^T {}^0R_7 \end{aligned}$$

Given the algebraic rotation matrix ${}^4R_7$ (Equation 3.35), derived from the homogeneous matrix ${}^4T_7$, and considering the global configuration parameter of the manipulator's wrist (GC6), joint θ_6 , followed by joints θ_5 and θ_7 , can be determined using Equations 3.36, 3.37 and 3.38, respectively.

$${}^4R_7 = \begin{bmatrix} \cos(\theta_5) \cos(\theta_6) \cos(\theta_7) - \sin(\theta_5) \sin(\theta_7) & -\cos(\theta_7) \sin(\theta_5) - \cos(\theta_5) \cos(\theta_6) \sin(\theta_7) & \cos(\theta_5) \sin(\theta_6) \\ \cos(\theta_5) \sin(\theta_7) + \cos(\theta_6) \cos(\theta_7) \sin(\theta_5) & \cos(\theta_5) \cos(\theta_7) - \cos(\theta_6) \sin(\theta_5) \sin(\theta_7) & \sin(\theta_5) \sin(\theta_6) \\ -\sin(\theta_6) \cos(\theta_7) & \sin(\theta_6) \sin(\theta_7) & \cos(\theta_6) \end{bmatrix} \quad 3.35$$

$$\theta_6 = \text{GC6} \arccos(a_{w33} \sin(\Psi) + b_{w33} \cos(\Psi) + c_{w33}) \quad 3.36$$

$$\theta_5 = \text{atan2}(\text{GC6} [a_{w23} \sin(\Psi) + b_{w23} \cos(\Psi) + c_{w23}], \text{GC6} [a_{w13} \sin(\Psi) + b_{w13} \cos(\Psi) + c_{w13}]) \quad 3.37$$

$$\theta_7 = \text{atan2}(\text{GC6} [a_{w32} \sin(\Psi) + b_{w32} \cos(\Psi) + c_{w32}], \text{GC6} [-a_{w31} \sin(\Psi) - b_{w31} \cos(\Psi) - c_{w31}]) \quad 3.38$$

In summary, it can be concluded that the calculation of the joints for the redundant 7-degree-of-freedom robot depends on the desired pose, as well as on the arm angle and the global configuration parameters.

3.4 Software Tools

For the development of this project, a set of software tools was utilized. Accordingly, this section begins with a brief introduction to the ROS2 framework. Next, it explores the MoveIt library, which is used for motion planning in robotic systems, establishes its relationship with the OMPL library and highlights Gazebo and RViz plugins. Finally, this chapter presents the physical simulator CoppeliaSim and the software Autodesk Fusion.

3.4.1 ROS2 Middleware

One of the most important tools to take into consideration in this work is the Robot Operating System (ROS) framework. ROS is an open-source framework consisting of communication tools and a set of plug-and-play libraries that aid in creating complex systems. ROS is considered a specialized middleware for robotics, providing communication infrastructure, hardware abstraction, low-level device control, exchange of messages between processes, and more (Serrano, 2015). Although it is independent of the type of programming language used, ROS primarily utilizes C++ and Python for its development (Figure 21).

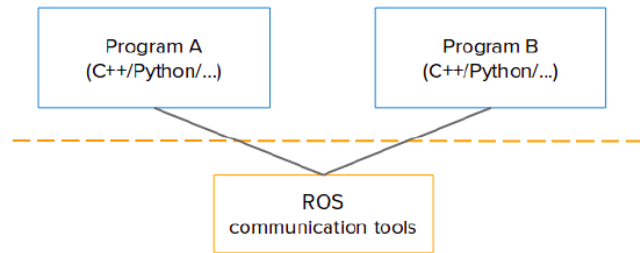


Figure 21 - Languages used in ROS communication tools. Adapted from (The Robotics Back-End, 2018).

ROS was created to facilitate code reuse in robotics research and development. It features an integrated repository that allows users to utilize pre-existing code for various robotic applications. ROS supports a wide range of robotic components, including aerial, ground, humanoid, and underwater robots, as well as different types of sensors, such as GPS receivers. As such, it serves as a platform that enables the integration of various software tools with different purposes, such as OpenCV and MoveIt. For instance, ROS can be integrated with OpenCV to enable data captured by cameras to be processed by advanced algorithms for more efficient image processing, such as object detection, motion tracking, and pattern recognition (Serrano, 2015).

In the MIAR Project, ROS was used to connect the selected motion planner for the redundant manipulator's movements in the logistics tasks and the physical software that serves as a robotic simulation environment. In this project, the operating system used is Ubuntu 22.04, and the recommended ROS distribution is ROS2 Humble Hawksbill. ROS2 Humble is an advanced version of the ROS framework.

Retaining some ROS concepts is important to ensure a correct understanding of the steps taken in the project in Chapter 4.

3.4.1.1 ROS Structure

ROS can be divided into workspaces containing packages. Each package can contain small programs called nodes, which enable connections between different systems.

3.4.1.1.1 ROS Workspace

A ROS workspace is a directory that starts empty and creates a specific structure when built using the Colcon build tool. It includes a source (src) subdirectory containing the source code for the ROS packages. Additionally, the workspace has three other directories: the build directory, where intermediate files are stored; the install directory, where all created packages

are installed; and the log directory, which holds various logging information about the builds made in the workspace.

3.4.1.1.2 Packages

Packages are the basic unit of software organization in ROS. Each package can contain libraries, executables, or scripts, which are used to structure and modularize the code. Packages can be created using either CMake or Python, allowing a workspace to include packages of different build types. Figure 22 illustrates a workspace structure containing a CMake package developed in C++ and a Python package developed in Python, both located in the source folder.

```
workspace_folder/  
  src/  
    cpp_package_1/          (CMake package)  
      CMakeLists.txt  
      include/cpp_package_1/  
      package.xml  
      src/  
  
    py_package_1/           (Python package)  
      package.xml  
      resource/py_package_1  
      setup.cfg  
      setup.py  
      py_package_1/  
      ...
```

Figure 22 - Workspace with CMake and Python packages.

3.4.1.1.3 Nodes

A node is an executable file that performs computation. Nodes are created inside packages. These processing units are independent and can communicate with other nodes. (Tavares, 2015). Figure 23 shows an example of a robotics application composed of nodes. This system consists of three ROS packages, each containing a set of nodes initiated separately and communicating with each other.

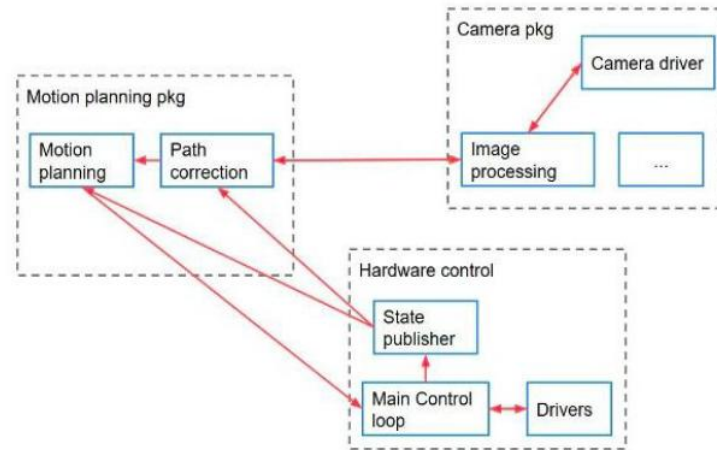


Figure 23 - Illustrative example of communication between nodes grouped into different packages. Adapted from (The Robotics Back-End, 2018).

3.4.1.2 Communication methods between nodes

Nodes can communicate with each other primarily through topics, services, and actions (Albonico et al., 2023).

3.4.1.2.1 Topics

Topics are used to send streams of data between nodes. Communication occurs through the sending and receiving of ROS messages by nodes via a topic. The node that sends/publishes a message to the topic is called a publisher node. Conversely, the node that subscribes to the topic to receive the respective sent message, i.e., to obtain the information, is called a subscriber node. There can be multiple publishers and subscribers simultaneously for a single topic, and a single node can publish and/or subscribe to multiple topics (Albonico et al., 2023). It is important to note that for this communication to occur, both the publisher and subscriber nodes must use the same message type. Messages are data structures that carry specific information. They can contain data such as numbers, strings, arrays, and other user-defined data types.

3.4.1.2.2 Services

Services are another method for nodes to communicate, allowing them to send requests and receive responses. Services are composed of two parts: a request and a response. In Figure 24, an example of a service is shown, where the inputs (requests) consist of two integer variables, and the output (response) is another variable called "result," also an integer.

```
int64 x
int64 y
...
int64 result
```

Figure 24 - Example of a srv file.

Communication with services is carried out by two nodes: the service client, which sends a message containing specific data that corresponds to the variables in the service request, and the service server, which interprets the data received in the request and responds with a message containing the results (the service response) (Albonico et al., 2023). It is also possible to associate multiple nodes with a service, meaning multiple service clients or servers can be involved. Figure 25 illustrates how communication is made by topics and services.

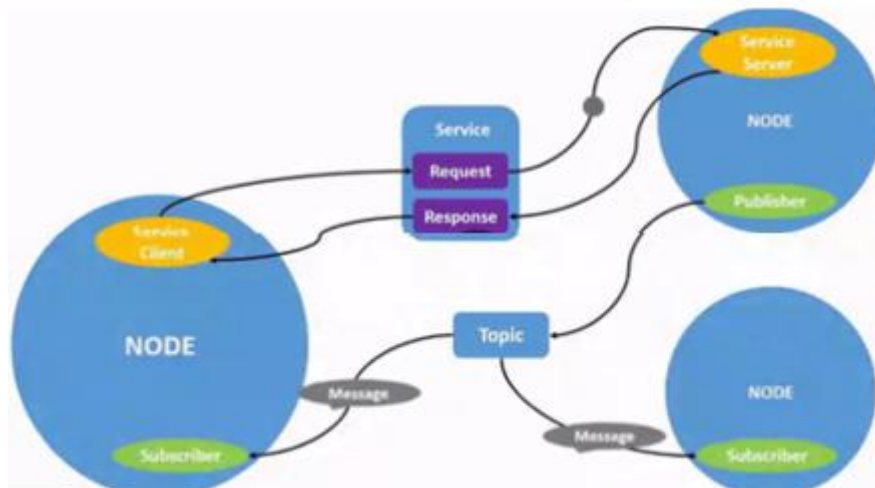


Figure 25 - Communication between nodes by a topic and a service. Adapted from (Open Robotics, 2024b).

When should services or topics be used? Topics should be used when unidirectional and continuous communication is required, meaning when there is a need for constant sending and receiving of a set of data. Services are necessary when performing remote procedure calls, that is, when a response to a certain request is needed. Services are used when the client makes a specific request and expects a specific response from the server.

3.4.1.2.3 Actions

Actions in ROS2 serve as a method of communication for tasks that take a long time to complete. An action consists of three main components: a goal service, a feedback topic, and

a result service. Like services, actions employ a bidirectional communication model through a client-server structure. The main difference is that, unlike services, which provide only one response, actions allow for continuous feedback and support the cancellation of orders.

When an action client submits a goal, the request is sent to an action server. The action server recognizes the goal and subsequently provides feedback and a final result. This process is illustrated in Figure 26.

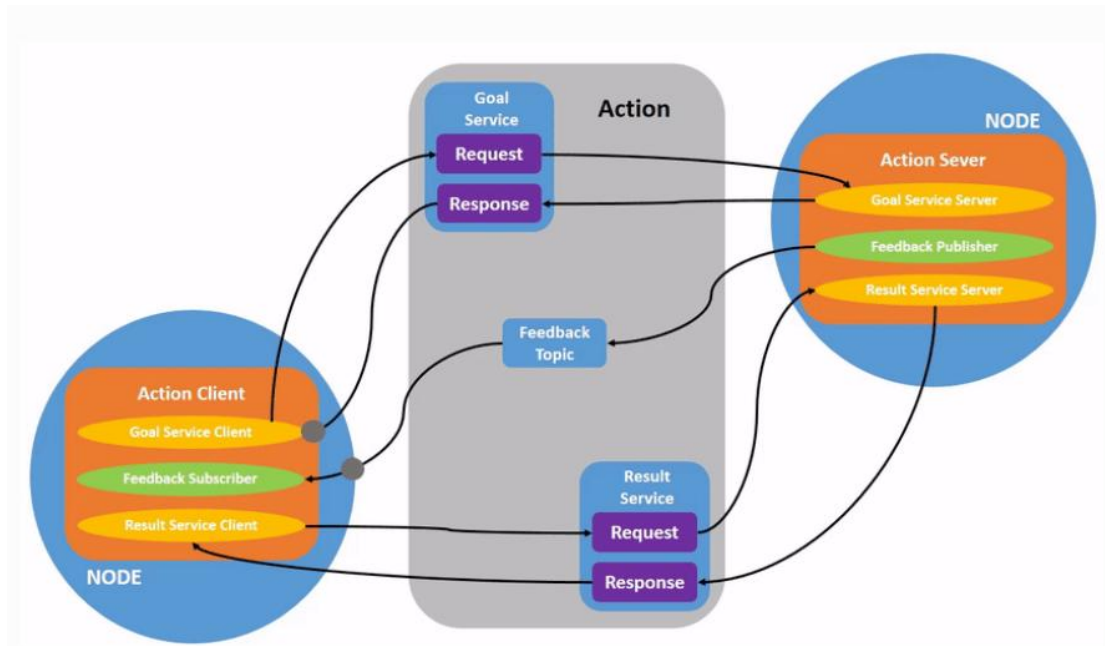


Figure 26 - Communication between nodes by an action. Adapted from (Open Robotics, 2024a).

3.4.1.3 Launch Files

Another important concept to consider in ROS is launch files, which play a crucial role in the configuration and initialization of robotic systems. As the name implies, launch files enable the simultaneous execution of multiple nodes rather than starting them one at a time. With launch files, users can initialize as many nodes as needed, including identical nodes simultaneously, and specify different priority levels for each node's execution. In addition to initializing multiple nodes, a launch file can also execute other launch files, which is why they are widely used in systems that control multiple robots.

3.4.2 MoveIt Library

Nowadays, there are a lot of tools available to solve path-planning problems. One of the most popular tools is the Open Motion Planning Library (OMPL) (I. A. Sucan et al., 2012). The OMPL is a C++ library that implements many state-of-the-art sampling-based planners, including PRM and RRT. The planners within this library are effective for systems with many degrees of freedom and differential constraints. However, with sampling-based methods, the user must specify the maximum time a given planner has to search for a solution once these methods cannot determine when a problem has no solution. It is also important to note that OMPL doesn't include functionalities associated with collision detection or visualization of the geometry of the workspace or the robot, so the user is responsible for choosing the necessary tools for this. Users are also responsible for defining collision detection methods or state validity checkers to ensure effective path planning. Therefore, given the mentioned limitations, OMPL is often integrated with other robotic frameworks like MoveIt Library, which is used for motion planning and can be interfaced through ROS packages.

MoveIt is a motion planning library built to work with the ROS middleware. This robotic manipulation platform provides kinematic calculation and collision checking and was designed to work with several motion planning algorithms or planning libraries and can be adapted to any ROS-supported robot (Coleman et al., 2014). In this project, MoveIt was used to validate the generated motion planning for the KUKA LBR iiwa 14 R820 manipulator during the pick-and-place operations of the *Targets*. One of the requirements of this project is to use ROS2 Humble, and since the operating system in use is Ubuntu 22.04, it is necessary that MoveIt version is compatible with it. In this case, the appropriate package for this ROS distro is the MoveIt 2 package. This package, `ros-humble-moveit`, is compatible with all sorts of robots, from robotic arms to mobile robots and even humanoids. This plugin will be used in the project to configure the robot to be controlled and plan and execute the robotic arm movements. To do so, it will be crucial to compute and optimize the trajectories of the robot arm from the given positions to the desired goals while respecting the arm's joint limits and avoiding obstacles.

Collision avoidance in MoveIt is achieved by the introduction of known obstacles in the environment before the planning process, allowing the algorithm to consider them and avoid creating trajectories that would lead to collisions. MoveIt handles collision checking for a

variety of objects, including meshes, where the objects are described by stl (standard triangle language) or dae (digital asset exchange) files and primitive shapes such as boxes, cylinders, cones, spheres, and planes (Chitta et al., 2012).

The selection of the right planner, the algorithm that calculates the motion plan, is a challenge since each planner has its strengths and weaknesses, and the best solution depends on the requirements of a specific task and the work environment (Moll et al., 2015). MoveIt provides many off-the-shelf planners, each with different approaches to generating robot trajectories. The default and most used planning library in MoveIt is the already mentioned Open Motion Planning Library (OMPL). This open-source planning library possesses various state-of-the-art sampling-based algorithms, including SBL, KPIECE, and their variants. Although MoveIt 2 currently supports OMPL, CHOMP, STOMP, and the planner PILZ, only OMPL planners are fully integrated into MoveIt, which is why they are primarily used in benchmarks. Thus, MoveIt includes a benchmarking package adapted for OMPL planners, which provides methods for comparing these algorithms and generating plot statistics using the OMPL Planner Arena.

In MoveIt, robots are specified by Unified Robot Description Format (URDF) files, which are the native format to describe robots in ROS. A URDF file is an XML specification to describe the robot's geometry and kinematics. These files contain the kinematic and dynamic description of the robot, the visual representation, and the collision model of the robot. The specification is represented in tree structures, where the robot consists of rigid links connected by physical joints. However, it is possible to add virtual joints to the URDF, for example, if it is necessary to represent the robot's root link's pose concerning a global coordinate system. The URDF is the most essential information for creating a MoveIt configuration package for a robot. A package can be created manually or generated automatically by the MoveIt Setup Assistant from a robot description file (usually the URDF). The MoveIt Setup Assistant is a GUI (Graphical User Interface) that helps set up any robot to work with MoveIt (Figure 27). This interface creates a MoveIt config package named `moveit_config` that contains the SRDF (Semantic Robot Description Format) file of the robot, which is a file that complements the URDF by defining joint groups, collision checking details, and any additional transforms required to describe the robot's pose fully. It also produces other configuration files (primarily YAML files) for the MoveIt pipeline, such as the `moveit_controllers.yaml`, which provides information about the available MoveIt controllers,

the joints associated with each controller, and the type of controller interface used; the `kinematics.yaml`, which specifies the inverse kinematics solvers for the robot's joints; `ros2_controllers.yaml`, which presents the hardware controllers that interact with the robot's joints and other devices; `joint_limits.yaml` that describes the different joint position, velocity and acceleration limits. This is useful for defining different joint limits without modifying the URDF file.

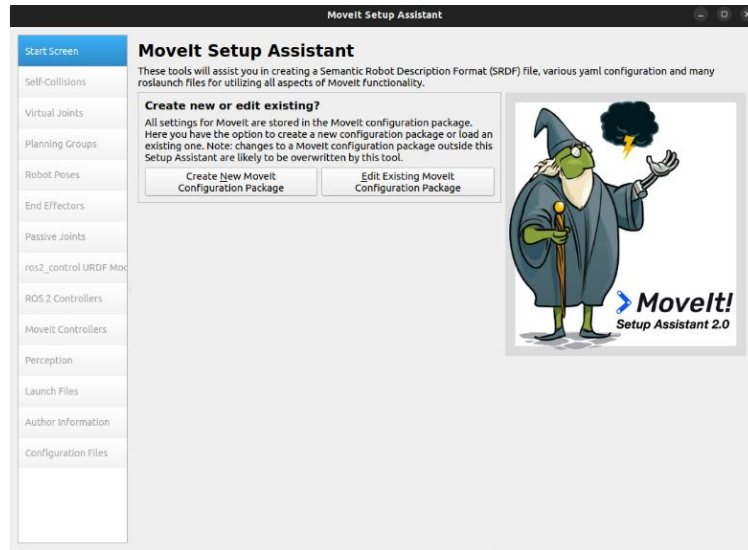


Figure 27 - Initial window of the MoveIt Setup Assistant. Adapted from (PickNik Robotics, 2024c).

The file related to the OMPL planners, `ompl_planning.yaml`, can also be added to the `moveit_config` package. This file includes settings related to the motion planning process, such as trajectory generation algorithms (e.g., RRT, PRM), planning time, and the number of attempts. These planners can be optimized by tuning the settings in this yaml file. The default values of these parameters are auto-generated in the MoveIt Setup Assistant. An example of a config folder created by the MoveIt Setup Assistant is shown in Figure 28.

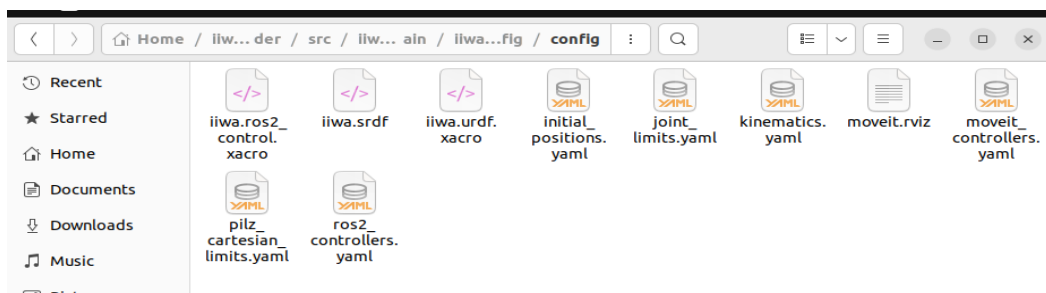


Figure 28 - Example of a config folder from the `moveit_config` package generated by the MoveIt Setup Assistant.

Figure 29 depicts the high-level system architecture of the `move_group` node, which is a central component in MoveIt. This ROS node operates as an integrator, coordinating different components to offer a set of ROS actions and services for users. There are three primary ways to access the services and actions of `move_group`: `move_group_interface` package, which provides a simple and effective C++ interface that communicates with the `move_group` node via ROS topics, services, and actions. It enables functionality for most tasks a user may need, such as defining the target's pose, motion planning, adding objects in the environment, and attaching or detaching objects from the robot; `moveit_commander` package through Python; Via a Graphical Interface, using the motion planning plugin for Rviz, the ROS visualization tool. Rviz is a 3D visualizer designed for the ROS environment (da Costa, 2019). It allows the display of sensor data and the robot's state in real-time, providing a virtual model of the robot. It presents sensor data through ROS topics and services, which helps understand the robot's perception of its environment. This simulator is mainly used to validate planning solutions in virtual environments, usually working in parallel with the physical software Gazebo, before being applied to the real robot. It enables visualization of the robot's trajectory, helping to identify potential issues. Gazebo is a 3D simulator designed to recreate environments for multiple robots. It includes dynamic and kinematic physics to accurately simulate real-world conditions such as friction, damping, and other environmental factors. The descriptive robot format used in Gazebo is the Simulation Description Format (SDF). As the Rviz, Gazebo is used to test and validate robotic behaviors before real operations. (Koenig & Howard, 2004)

The `move_group` retrieves different types of information from the ROS parameter server: the `robot_description` parameter, which provides the robot's URDF file, and the `robot_description_semantic` parameter, which provides the robot's SRDF file. It also searches for other MoveIt configuration data like joint limits, kinematics, motion planning settings, and perception information. These config files are also generated within the MoveIt config package of the robot by the MoveIt Setup Assistant.

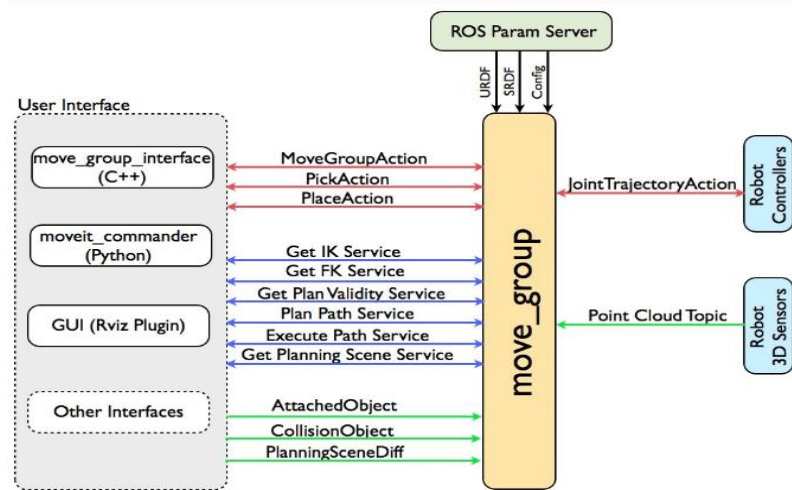


Figure 29 - High-level system architecture of the MoveIt `move_group` node. Adapted from (PickNik Robotics, 2024a).

In MoveIt, even if the trajectory has already been planned for a given task, it still needs to be executed. To do so, MoveIt uses controllers to execute the trajectory for certain hardware. The `ros2_control` framework standardizes the interaction between the robot hardware and the software controllers (Amadi et al., 2024). It provides a flexible interface to control different types of hardware, whether for real robots, physical simulations, or mock setups. This framework comprises the controller manager node, the controllers required for trajectory execution, and hardware interfaces. The controller manager node manages controller switching by activating, deactivating, and loading controllers as needed. It connects the controllers to the required hardware interfaces and manages MoveIt by activating the appropriate controllers. The users can also access the controller manager through ROS services. The controllers are plugins based on control theory that evaluate input commands and calculate outputs to control robot joints. The most used controller plugins are the `joint_trajectory_controller`, which controls the trajectories of robot joints; the `joint_state_controller`, which publishes the status of robot joints; and the position, velocity, and effort controllers, which send the desired position, velocity, and effort commands, respectively, to the hardware interface. As mentioned earlier, these controllers are configured in the `ros2_controllers.yaml` file. The hardware interface is used to send and receive commands from the hardware. There are several hardware interfaces, the most common being the Joint Command Interface, which supports joint control, and the Joint State Interface, which supports reading the state of the joints (position, velocity, effort). In case the hardware interface of a robot in MoveIt doesn't already exist, the user can create its own.

Several hardware plugins are available regarding the hardware with which MoveIt can communicate. Among these stands the real hardware of a robot that corresponds to the robot's system interface and the `gazebo_ros2_control/GazeboSystem`, which integrates the control of the robot with the Gazebo simulator, allowing realistic physical simulations. If the goal is to simulate the robot's behavior without relying on a physical simulator or the real robot, fake hardware can be used in MoveIt through the `mock_components/GenericSystem` plugin. This plugin is mainly used for offline testing of the `ros2_control` framework, as it simulates ideal robot behavior by mirroring commands to their states. Since in the project, the validation tests of the trajectories generated from the manipulator were not to be supported by the Gazebo physical simulator, then the `GenericSystem` plugin was used.

To visualize a robot in MoveIt, as well as plan and execute its motions, ROS 2 launch files are required. The Python launch files handle the loading of all the configuration parameters present in the `moveit_config` package. To achieve this, MoveIt uses the `MoveItConfigsBuilder` utility to load these parameters, which include the robot description and semantic description files (URDF and SRDF), motion planning and kinematics plugins, trajectory execution settings, and other necessary configurations. Once the MoveIt configuration parameters are loaded, the `move_group` node must be launched to initiate the Move Group C++ Interface. It's also important to launch the `tf2` library to manage coordinate transforms, especially the static transform publisher node, which establishes the static transform between the robot's base and world frames. The robot state publisher node must be launched to read the robot's joint states, compute frame transformations using the URDF file, and publish these transforms to `tf2`. Finally, the `ros2_control` framework must be set up for the robot to execute a trajectory. To start the `ros2_control`, the controller manager node must be launched and spawn the required controllers and the hardware plugin needed for the trajectory execution.

3.4.3 CoppeliaSim

Although MoveIt does perform motion planning for the manipulator for the various pick-and-place operations, it is essential to validate the planning tests before applying them to real-world scenarios. To do this, the movements must first be simulated in a dynamic analysis. While Gazebo is the simulator that typically works alongside MoveIt, this project will utilize CoppeliaSim to verify the arm's movements in a digital environment.

CoppeliaSim (Figure 30) is a physical simulator that aids the development and validation of planning algorithms and robotic systems (Bogaerts et al., 2020). This software allows the creation of any robot and the design of complex environments with multiple robots. It includes a physical engine that enables robotic behaviors that are more similar to real solutions.

The selection of CoppeliaSim for this project is due to its compatibility with various operating systems, including Windows and Linux, making it suitable to work in Ubuntu 22.04, which is the required version for the project. CoppeliaSim also supports the creation of robotic models in the working environment by importing CAD files such as stl, and can also read URDF files like in MoveIt. This compatibility allows the same robot model to be used in both MoveIt and the physical simulator. Furthermore, CoppeliaSim can also establish external communication with other software because it's compatible with ROS. These strengths make it a versatile tool for connecting the motion planning of the robotic arm in MoveIt with the robot's physical model in CoppeliaSim.

CoppeliaSim is organized into models, which are individual elements that can represent a robot, sensors, or even a person, and a scenario, which comprises all the models that compose the simulation space. Each model is represented by a hierarchy tree that includes the physical model, which considers all the physical properties, such as mass and moments of inertia, and the graphical model, which is associated with the visual representation of the model in the scenario. The tree can also have joints connecting the model with any other scenario component. CoppeliaSim allows the control of different models by associating them with child scripts. The proprietary language for these scripts is Lua or Python.

In this work, CoppeliaSim replicates the manipulator's movement during the pick-and-place operations through ROS and validates the planned motions by recreating a digital twin of the robot's workspace.

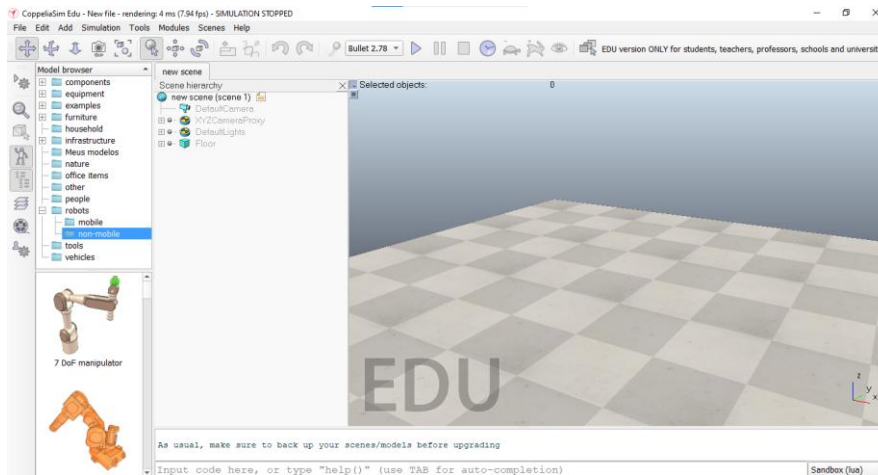


Figure 30 - CoppeliaSim interface.

3.4.4 Autodesk Fusion

Lastly, Autodesk Fusion was used, which is a Computer-Aided Design (CAD), Computer-Aided Engineering (CAE), and Computer-Aided Manufacturing (CAM) software developed by Autodesk. This software (Figure 31) allows users to define and model various components, set properties, and assemble different models. It also helps validate products by subjecting them to statistical, thermal, dynamic, and other analyses. In the context of this work, Autodesk Fusion was utilized to model the access ramps and any other essential components for the robot's workspace during the planning of the pick-and-place operations. Additionally, the software was used to simulate the performance of the custom fingers designed for the selected gripper of the robot, as detailed in Section 4.4.2.

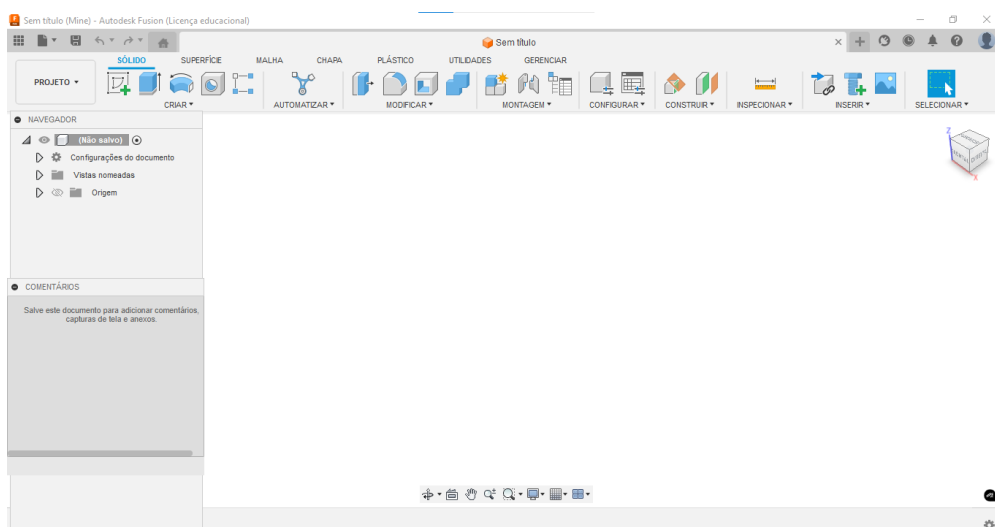


Figure 31 - Autodesk Fusion Interface.

4 Developed Work

The present chapter encompasses the practical elements undertaken in the development of this project. The chapter begins with a description of the system architecture of the MIAR project, emphasizing the module where this dissertation fits. It then addresses the creation of the simulation model of the robotic arm's workspace and the selection of a suitable gripper for the desired operations, including the development of customized gripper fingers. Furthermore, it outlines the configuration and control of the set: manipulator and gripper for Movelt and the integration with the CoppeliaSim software. The last section exposes the implementation of motion algorithms for the robotic arm's pick-and-place operations.

4.1 System Architecture

As previously indicated, this dissertation is part of the MIAR project, whose objective is to transport *Targets* from an intermediate storage buffer to supply ramps. Thus, developing a system architecture encompassing several modules was necessary. Figure 32 illustrates the global architecture of the MIAR system.

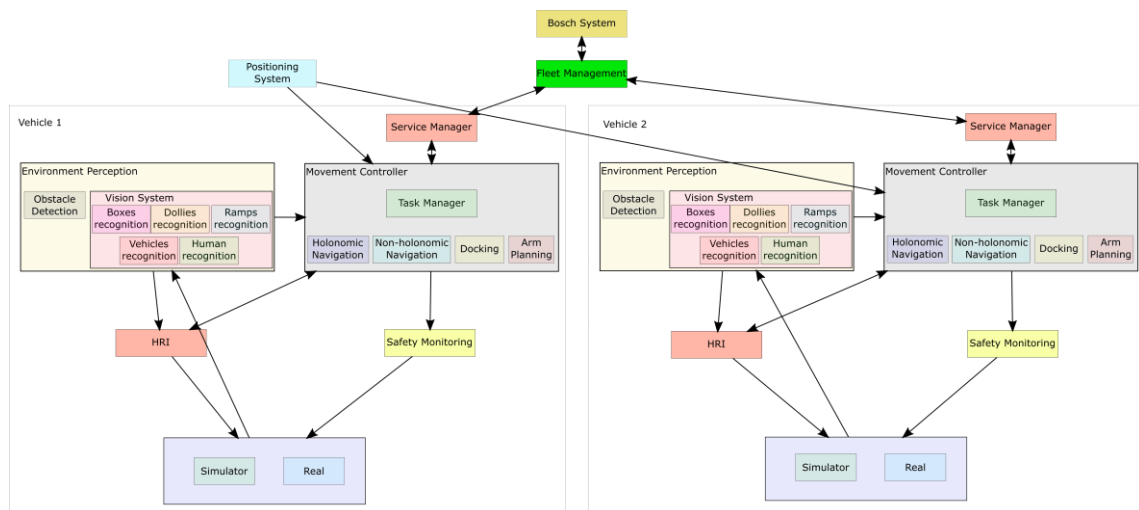


Figure 32 - Global architecture of the MIAR system.

This architecture is centered around the core module, Fleet Management, which is responsible for managing the fleet of vehicles and contains a map of the production lines' supply area. This module receives data from the production lines through the Bosch system

and transmits the information to each vehicle via the local module Service Manager. This module receives the service of the Fleet Management and divides it into tasks that are sent to the Task Manager. This submodule is within the Movement Controller Module, which includes several submodules dedicated to controlling the AMR and planning the manipulator's movements. The task manager ensures the proper execution of assigned tasks by activating the appropriate submodules in the correct sequence. The architecture also includes the Environment Perception local module that comprises several submodules responsible for acquiring and processing sensory information, either through LiDAR or the vision system. This dissertation focuses primarily on the Arm Planning submodule, which is tasked with planning and controlling the arm's movements. However, developing this submodule requires understanding the mobile manipulator's conditions. This includes a clear definition of the tasks to be performed by the mobile manipulator and its workspace during the operations.

4.2 Simulation Model

As previously mentioned, the main aim of this work is to develop safe and smooth trajectories for the KUKA LBR iiwa 14 R820 manipulator. The respective arm is mounted on the KUKA KMR iiwa 14 mobile platform. In the Bosch Car Multimédia BRgP shop floor, the *Targets* are positioned on top of dollies, which are transport carts used to move loads within the facility. The *Targets* are manually transferred from the dollies to different assembly lines by PoUPs, a designation established for the operators responsible for the specific task. The objective of the MIAR project's "To Be" challenge is to replace these PoUPs with autonomous mobile manipulator robots to perform the task. An example of the intended work layout is shown in Figure 33.

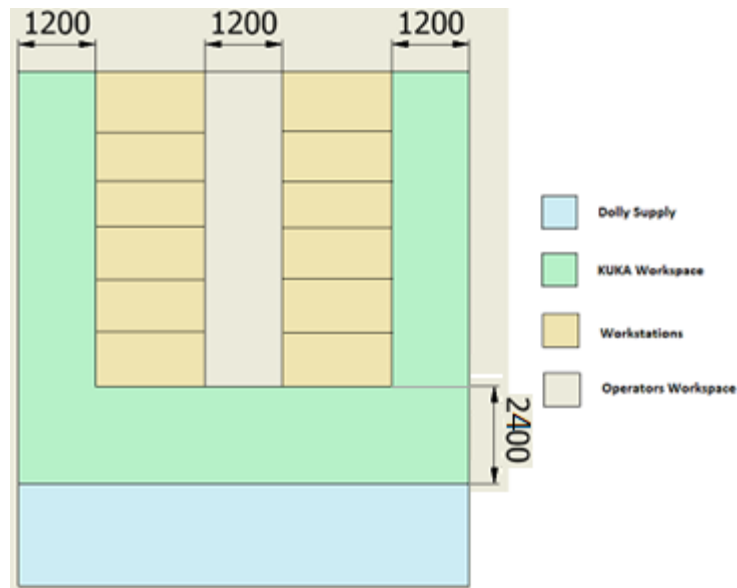


Figure 33 - Intended work layout at Bosch Car Multimedia BRgP.

As can be seen in Figure 33, the blue zone corresponds to the initial location of the dollies loaded with their respective *Targets*.

First, the mobile platform must move to the Dolly Supply zone so that the robotic arm can pick up the *Targets* and place them on top of the platform. Afterward, the platform moves to the different workstations, represented in yellow in Figure 33, allowing the arm to deliver the *Targets* to the appropriate stations. It's important to note that the corridors where the autonomous mobile manipulator robots operate are 1.2 meters wide, which limits their movement during the execution of the desired tasks. The corridor next to Dolly Supply is wider, allowing mobile platforms to rotate when picking up boxes.

Several preliminary parameters must be considered to carry out the simulation model, namely the maximum number of *Targets* positioned on top of the dollies that the arm can reach and the maximum number of *Targets* that the mobile platform can transport from the Dolly Supply to the different workstations. The selection of the gripper for the robotic arm is also extremely important for the manipulation of the *Targets*.

Thereby, some fundamental elements required to recreate the intended work layout at the Bosch Car Multimedia BRgP shop floor were modeled using Autodesk Fusion software. These elements include the access ramps and the dolly channel. Other essential components, like the *Targets* and the dollies, were provided as CAD files by Bosch.

There are two types of *Targets* to be manipulated, both of which are plastic PP containers from the Utz Group (Utz Group, 2014). One container is larger, measuring 600 x 400 x 220 mm, while the other is smaller, measuring 400 x 300 x 220 mm. Both containers have loose lids. Figure 34 presents these containers and their corresponding CAD models.

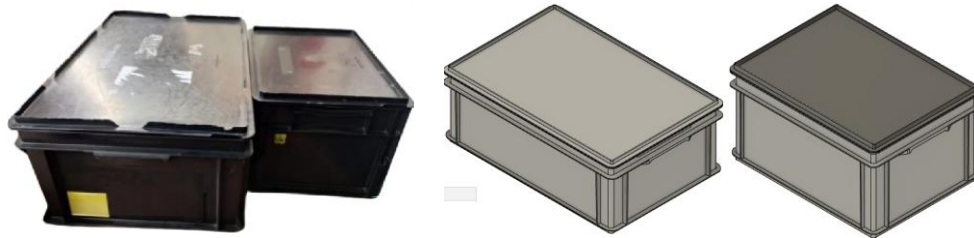


Figure 34 – Containers-RAKO from Utz Group and their respective CAD models.

In this dissertation, the larger containers will be referred to as "600x400 boxes" and the smaller ones as "400x300 boxes" based on the outer dimensions of their bases (Figure 35).

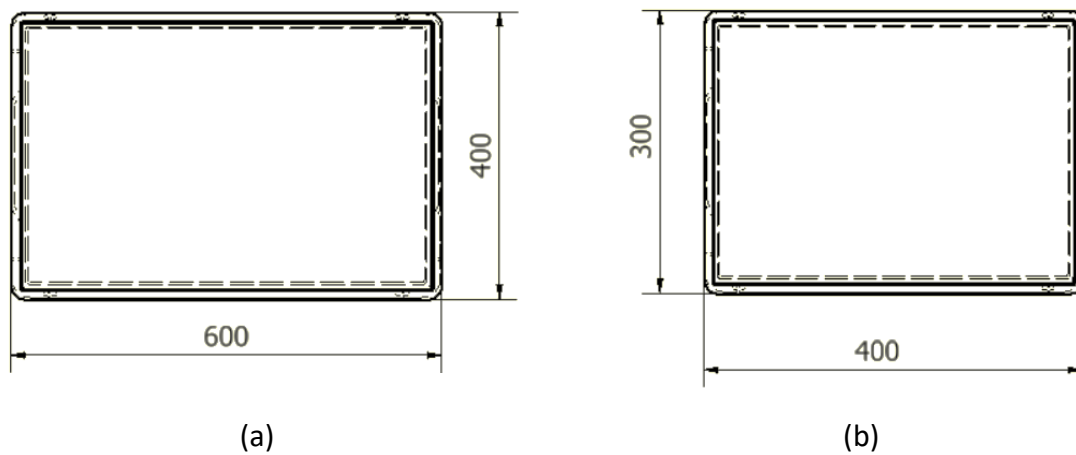


Figure 35 - (a) Outer dimensions of the base of the larger container; (b) Outer dimensions of the base of the smaller container.

Similar to the *Targets*, there are two types of dollies, each designed for a specific kind of container. Currently, the dollies are positioned together at the Dolly Supply without any secure mechanism to hold them in place (Figure 36).



Figure 36 - Dolly Supply zone for MIAR Project.

As a result, the dolly channels are used to retain and guide the respective dollies. An example of this mechanism, along with its CAD model, can be seen in Figure 37.



Figure 37 – Dolly Channel and CAD representation.

Finally, the access ramps are responsible for storing the boxes for a certain period of time. In the current shop floor, the ramps have either two levels, which include an upper and a lower level, referred to as one-level ramps, or three levels, consisting of an upper, middle, and lower level, referred to as two-level ramps. The upper and middle levels are designated for holding full boxes from the Dolly Supply provided by the PoUP, while the lower level contains the empty boxes left by the human operators. The CAD models of the one and two-level ramps can be checked in Figure 38 and Figure 39, respectively. There are two types of ramps corresponding to each kind of *Target*, where the width of each level of the ramp is the box length with an additional 10 mm, implying high precision in the picking and placement of the boxes. Appendix A shows these four ramps without the roller rails in more detail.

For this project, the one-level ramps will be used for the pick-and-place operations. Thus, the mobile manipulator has to place the boxes retrieved from the Dolly Supply on the upper level of the ramps, pick the empty boxes from the lower levels, and transport them back to the Dolly Supply.



Figure 38 - One-level access ramp for the storage of *Targets*.



Figure 39 - Two-level access ramp for the storage of *Targets*.

Using the CoppeliaSim software, a complete scenario of the work layout in which the mobile manipulator must operate was created (Figure 40). However, since this work focuses solely on controlling the redundant arm during the manipulation tasks and to avoid high computational costs when running the simulation due to the large number of elements present, the environment in CoppeliaSim was limited to a single workstation for this dissertation. So, six manipulation scenarios were considered: two comprising the ramps for the 600x400 boxes, another two with the ramps for the 400x300 boxes, and two containing a small sample of the Dolly Supply for each type of box, which were used for the arm to place the boxes from and to the platform.

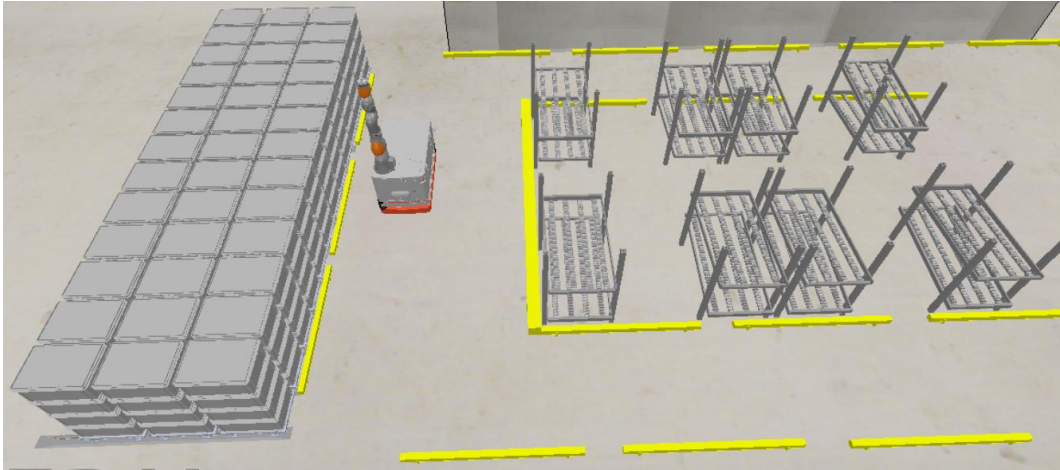


Figure 40 - Representation of the work layout using the CoppeliaSim software.

4.3 Selection of the Gripper

To plan the trajectories for simulation and validation in an initial phase before moving to a real case, a gripper is needed for the robotic arm to grasp the objects. Although the robotic arm intended for the MIAR project has already been defined, a suitable gripper for manipulation has not yet been considered.

Due to the temporal limitations in completing this dissertation and since this work focuses on developing trajectories for the manipulator rather than designing a gripper, the gripper selection was based on available off-the-shelf solutions. The goal was to select a gripper that best fits the project's requirements for this initial phase. It is important to emphasize that the gripper discussed in this section is not necessarily the final solution for the entire project. Instead, it represents an initial choice that enables the analysis of motion planning, both in simulation and subsequent real-world applications.

Several gripper manufacturers exist, including Festo, OnRobot, Robotiq, Schunk, Zimmer, and SMC. These companies offer a wide portfolio of robotic tool solutions that utilize different grasping principles, such as impactive (which includes angular, parallel, and radial) and astrictive (magnet and vacuum) approaches.

The selection of a gripper is normally based on expert knowledge. Therefore, choosing the appropriate one should follow a rational procedure. To simplify this, an objectives tree was made to clarify all the goals to reach by dividing them into different levels.

MIAR Project Gripper

- Manipulation of objects
 - Capability to grasp/ungrasp the *Targets*
- Data Handling
 - Sending and receiving data in real-time
 - Compatibility with different software platforms
- Flexible Solution
 - Safety
 - Safe for collaborative operations
 - Robustness
 - Mechanical resistance
 - Adaptability
 - Compatible with different robotic arms

As the lid of the *Targets* is loose and the boxes have dimensions in the order of 10^2 mm, parallel grippers are intended to be used. The gripper must be capable of grasping both types of *Targets*, ensuring that the gripping side matches the length of the boxes. As a result, the distance between the gripper fingers must be greater than 400 mm, corresponding to the smallest *Target's* length. Additionally, the maximum weight that the *Targets* can hold is 8 kg. Since the Kuka LBR iiwa 14 R820 collaborative arm has a payload capacity of only 14 kg, the selected gripper must be limited to a maximum weight of 6 kg.

Table 3 was created to clarify the gripper's requirements. This table is divided into functional requirements, which describe what the system must do to meet the project objectives, and non-functional requirements, which describe how the system should work and specify the system's quality characteristics. The gripper's height should be as low as possible because this variable directly affects the arm's payload.

Table 3 - Functional and non-functional requirements for MIAR project gripper.

Functional Requirements
Stroke ≥ 200 mm
Payload > 8 kg
Weight ≤ 6 kg
Minimum distance between teeth > 400 mm
Handle 600x400 boxes and 400x300 boxes
Non-functional Requirements
Integration with ROS2 (software compatibility)
Collaborative
Lightweight
Easy control
High precision
Small gripper height
High Payload

Based on the mentioned requirements, the grippers available in the market were analyzed. Table 4 presents the main attributes of the chosen solutions that best match the intended requirements.

Table 4 - Comparison of product attributes.

	1	2	3	4	5
Solution	PE25200 - Gimatic 	LEG 400-2- 15-2-10X3-B - SCHUNK 	LEG 520-2- 15-2-10X3-B - SCHUNK 	LEG 760-2- 15-2-10X3-B - SCHUNK 	Onrobot – 2FGP20 (base unit) 
Stroke (mm)	200	202	322	562	260
Weight (kg)	2.08	5.4	6.4	7.9	3.7
Payload (kg)	-	7.5	7.5	7.5	20
Max. Gripping Force (N)	460	1500	1500	1500	400

Only the PE25200 Gimatic is pneumatically actuated among the selected grippers, while the others are electric. Only the 2FGP20 is classified as a collaborative gripper, whereas the others are considered industrial grippers. This distinction can be easily confirmed by comparing the payload of each solution. It is important to clarify that the payload listed in Table 4 for the Schunk grippers represents the payload for the recommended workpiece weight to be manipulated, whereas the payload for 2FGP20 is the maximum allowable value.

Since some performance attributes are more important than others, the selection of the right gripper solution was guided by the weighted objectives method, which is a way to evaluate and compare different solutions. This method establishes criteria based on the initial objectives to reach. Some criteria can be more general, while others may be more specific regarding the problem at hand. The comparison criteria are then ranked and assigned by numerical weights. The criteria were systematically compared, assigning a value of 1 to the most important one. The importance order can be checked in Figure 41.

	ROS2 compatibility	Control	Weight	Load capacity	Safety	Stroke	Sum
ROS2 compatibility		0	0	0	0	0	0
Control	1		0	0	0	0	1
Weight	1	1		1	1	0	4
Load capacity	1	1	0		1	0	3
Safety	1	1	0	0		0	2
Stroke	1	1	1	1	1		5

Figure 41 - Systematic Comparison between criteria.

Subsequently, the assignment of relative weights to the sorted criteria was made, as shown in Figure 42.

Comparison Criteria	Scale	Weighted score
Stroke	5	0,33
Weight	4	0,27
Safety	3	0,20
Control	2	0,13
ROS2 compatibility	1	0,07

Figure 42 - Attribution of weights relative to criteria.

Utility scores must be established to compare the chosen solutions against the considered criteria. These scores were settled in four classes: 0 -> inappropriate; 1 -> weak; 2 -> Satisfactory; 3 -> Good; and 4 -> Excellent. Applying these scores to the solutions made it possible to choose a suitable gripper for the MIAR Project (Figure 43).

Criteria	Weighted score	Solution				
		1	2	3	4	5
Stroke	0,33	0	1	4	4	3
Weight	0,27	4	1	0	0	2
Safety	0,20	2	2	2	2	3
Control	0,13	3	3	3	3	2
ROS2 compatibility	0,07	0	0	0	0	3
Total		1,87	1,40	2,13	2,13	2,60

Figure 43 - Weighted objectives evaluation chart for five gripper solutions.

Based on the applied method, the most appropriate solution was the 2FGP20 gripper from OnRobot (Figure 44). This gripper can communicate using ROS, which is beneficial for integrating with the motion planning software MoveIt, which is responsible for planning the robot's trajectories.

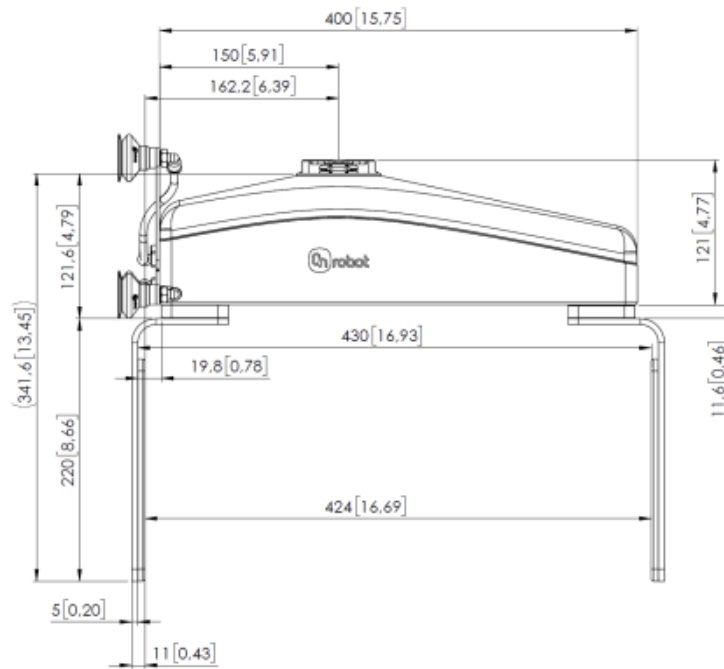


Figure 44 - 2FGP20 front view (dimensions in mm). Adapted from (OnRobot, 2023).

4.4 Development of the Gripper Fingers

As stated in Section 4.3, one of the requirements is that the gripper must have a minimum distance between its fingers greater than 400 mm to grasp the boxes successfully. Although the stroke of the 2FGP20 gripper is 260 mm, its opening is insufficient for grabbing the *Targets*. To ensure effective manipulation of both boxes, it is essential to develop custom gripper fingers. Using Autodesk Fusion software, the desired fingers were modeled, and through a Finite Element Analysis (FEA), the fingers were tested to verify whether they could meet the requirements to be used in a real application. The FEA method is useful for simulating and identifying causes of failure in mechanical systems to prevent failures before manufacturing.

4.4.1 Modeling the Gripper Fingers

Typically, parallel gripper fingers are composed of simple plates that apply pressure to the manipulated object. The tangential component of the applied pressure must be sufficient to reach a static equilibrium, balancing the gravitational force and preventing the object from falling. However, the boxes under study feature a groove corresponding to their outermost surface, which has a thickness of only 6 mm, as can be seen in Figure 45, which illustrates the profile view of both *Targets*.

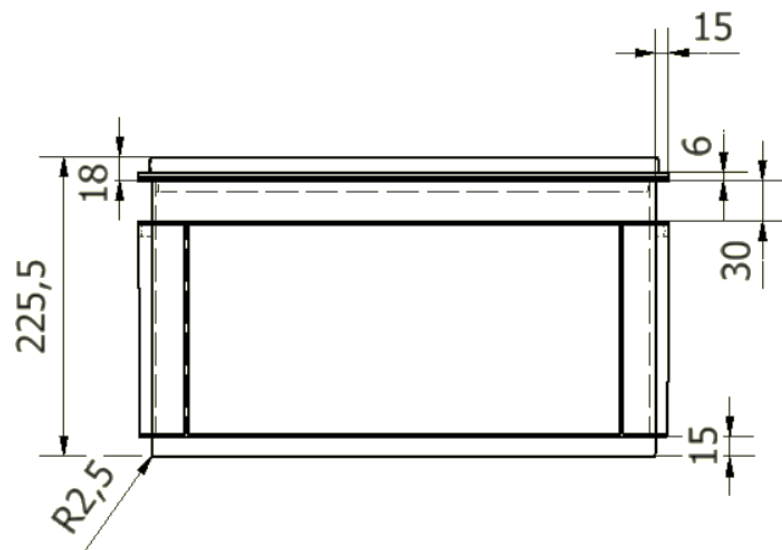


Figure 45 - Profile view of both *Targets* with the outer dimensions.

To address this, the gripper fingers were inspired by how a human hand typically grasps these types of boxes. The fingers were designed in an L shape, allowing them to compress and handle the boxes, as shown in Figure 47. This design approach reduces the pressure required on the outer surfaces of the box by applying load components in both the horizontal and vertical directions. As detailed in Figure 45, the groove width is 15 mm, which means the minimum aperture for the 400x300 boxes must be 370 mm. To ensure a safe grip, the design of the fingers allows a distance of 368 mm when the gripper is closed. This design enables the gripper to clamp the boxes in 2 mm if necessary (Figure 46). When the gripper is open, the distance between the fingers reaches 628 mm because the gripper's stroke is 260 mm. This results in a total clearance of 28 mm in relation to the 600x400 boxes. Consequently, high precision is required when planning the grasping and releasing operations of the robot. The dimensions of each finger can be verified in Appendix B. In the CAD model, fillets were applied

to the perpendicular surfaces where the boxes are grasped, aiming to reduce stress concentrations and facilitate mesh generation.

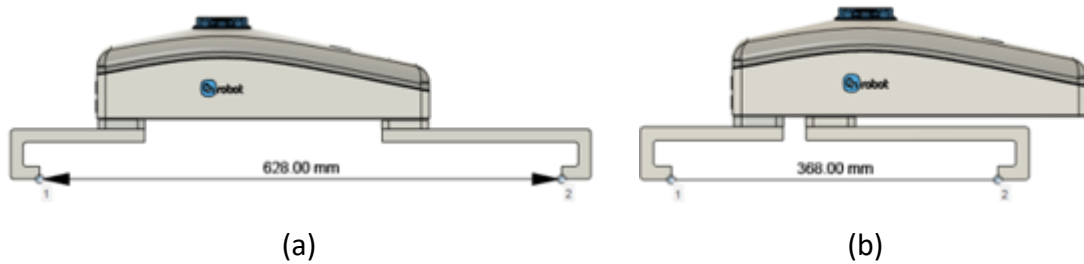


Figure 46 – (a) Maximum Gripper Aperture; (b) Minimum Gripper Aperture.

One of the main problems of the 2FGP20 gripper is that its flange is decentralized 50 mm in relation to its center, as reported in Figure 44. Another issue is that only one of the gripper fingers is movable. Since the two boxes to be manipulated have a length difference of 200 mm if both fingers have the same length, a significant misalignment would occur between the gripper's flange and the center of one of the boxes. Such misalignment is undesirable, as it could create a moment during manipulation tasks that might compromise the robot's operations. To mitigate this problem, the fingers must have different sizes. Therefore, the length of each finger was adjusted interactively until reasonable dimensions were achieved. This ensures that the distance from the robot flange to the center of each box isn't too different. Since these fingers are intended for initial tests, one of the Bosch requirements was that they should be made of PLA.



Figure 47 - Gripper holding the two types of *Targets*.

4.4.2 Evaluation and simulation of the fingers

After creating the geometric model of the fingers, the pre-processing of a linear static analysis was made, where inputs like the materials, mesh, and loading conditions were

established. Regarding loads, the fingers were subjected to two forces: a horizontal force representing the maximum gripping force of 400 N and a vertical force corresponding to the maximum weight of the *Targets*. This latter force is responsible for provoking tensile stresses in the upper surface of the fingers and compressive stresses in the lower surface. Since the two fingers withstand the boxes of 8 kg, an acting force of 39.24N was applied to the inner surfaces of each finger. Regarding boundary conditions, each finger is fixed to the claws at the base of the gripper, which constrains all six degrees of freedom (DoF) of the model.

The mesh is a crucial factor to consider, as its quality significantly impacts the simulation results. A higher mesh density, which means more elements by unit of volume, leads to more accurate results. However, higher computation costs will be required if the mesh is too refined. Since each finger is voluminous, solid elements were considered, specifically quadratic tetrahedral elements. These elements are composed of 4 corner nodes, 6 mid-side nodes, and 6 edges. Second-order elements generally yield more realistic results than linear elements because they can have curved edges to better capture geometric curvatures. This feature was considered in the mesh generation (Figure 48).

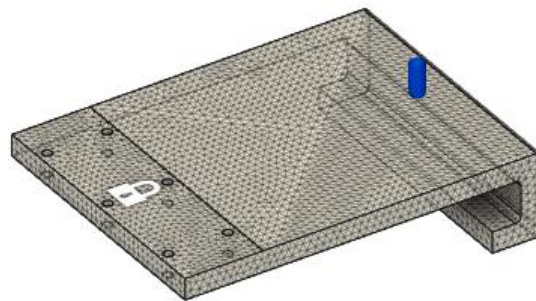


Figure 48 - Pre-processing of the mobile gripper finger.

Instead of using local mesh refinement, where local mesh controls are applied to different regions of the model, an adaptive mesh refinement was used. In this method, the solver automatically identifies critical stress regions and refines the mesh by employing smaller elements in those areas. As the mesh density increases, the stress levels tend to rise until the results stabilize. From that point, further mesh refinement does not yield significant differences in the results. A convergence study was conducted to achieve what is known as a converged mesh. This method involves successively refining the mesh until the change in a critical parameter, usually stress, between two consecutive iterations is less than a certain

percentage (Madier, 2021). In this case, a percentage of 5% was set. To avoid excessive computational times during the mesh generation, a maximum of 6 iterations was defined (Figure 49).

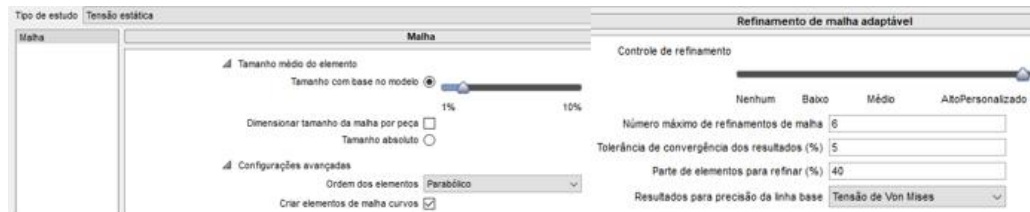


Figure 49 - Mesh generation parameters in Autodesk Fusion.

Following pre-processing, the fingers were subjected to static analysis to determine their ability to support the applied loads during the grasping actions. Due to the high applied loads, a safety factor of 4:1 based on critical transportation was considered for comparison (PHD Inc., 2001). In the post-processing phase, the results were analyzed. Based on the results obtained, it can be concluded that the fingers can withstand the applied loads, as the values for stress, strain, and displacement are low, and the safety factor is high throughout the part. As expected, the maximum stress and strain occur on the perpendicular internal surfaces due to the moment generated by the gripping force. The maximum displacement is observed at the tips of the fingers, as this gripping force is significantly greater than the weight of the box. Moreover, the stress on the movable finger (Figure 50 – (a)) is slightly higher than that on the fixed finger (Figure 50 – (b)) because the movable finger is longer, resulting in a greater moment at its tip.

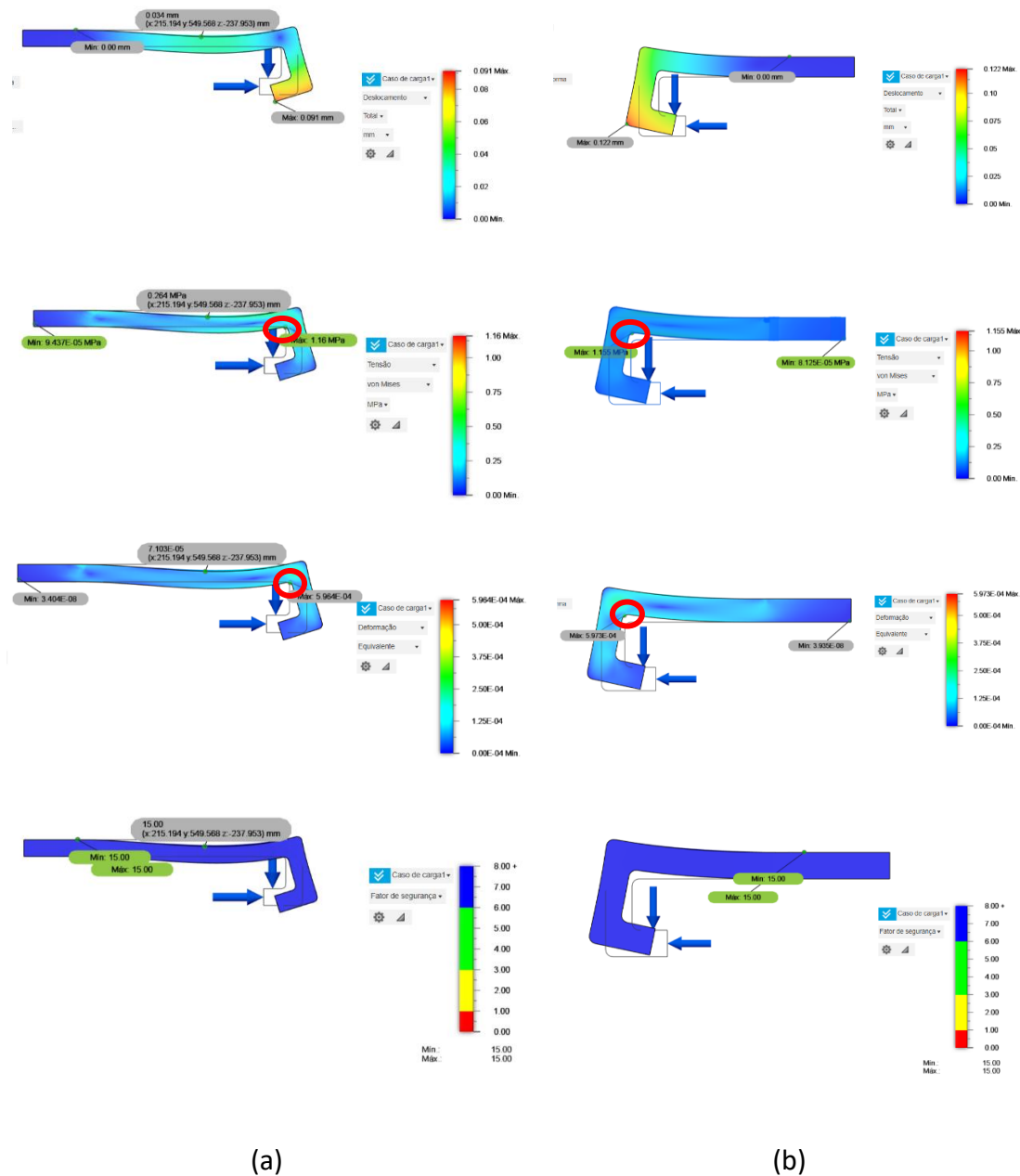


Figure 50 – (a) Static simulation on the movable finger; (b) Static simulation on the fixed finger.

Given the high safety factor, it's possible to understand that the fingers were over-dimensioned. The design can be optimized to reduce the weight of the fingers. A shape optimization was considered, as shown in Figure 51. However, since these fingers were intended solely for initial testing, this study has not been conducted. Therefore, the fingers used were those described earlier, with a total weight of 1 kg.

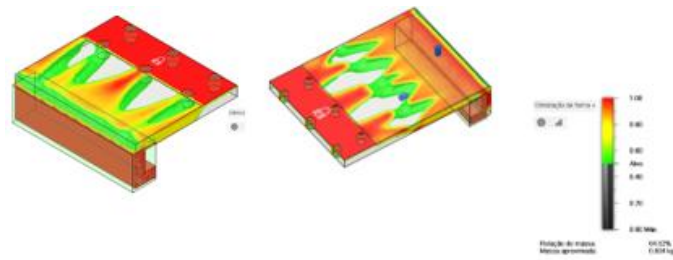


Figure 51 - Fingers shape optimization.

For safety precautions, the study was also conducted on the boxes to determine if they could withstand the gripper's maximum gripping force, which can reach high levels. As anticipated, the maximum gripping force did not impact the boxes (Figure 52). Given that the maximum experienced stress is 3.1 MPa, it can be concluded that the box remains within the elastic regime and doesn't suffer plastic deformation because the box has a yield stress of 30.30 MPa.

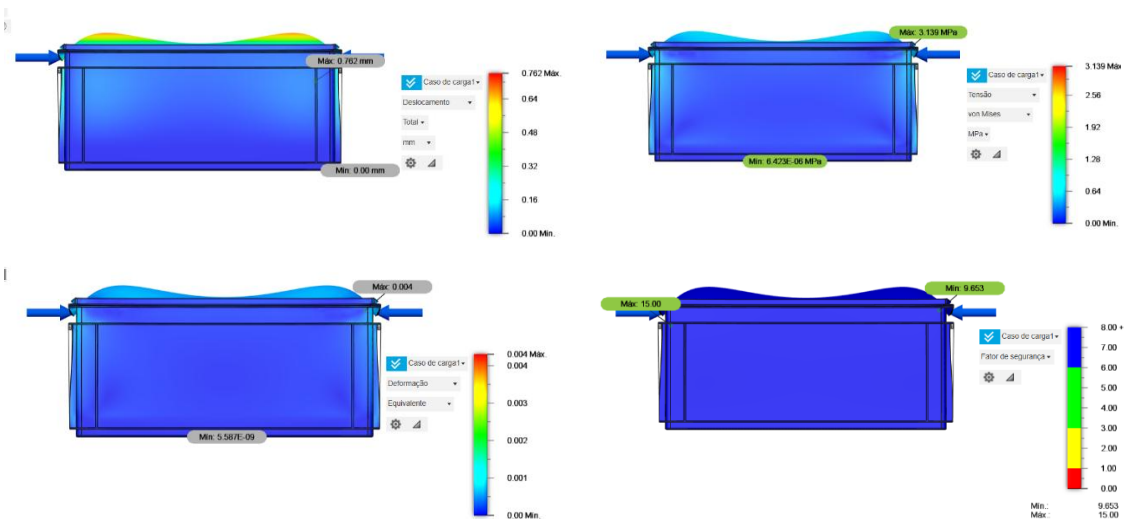


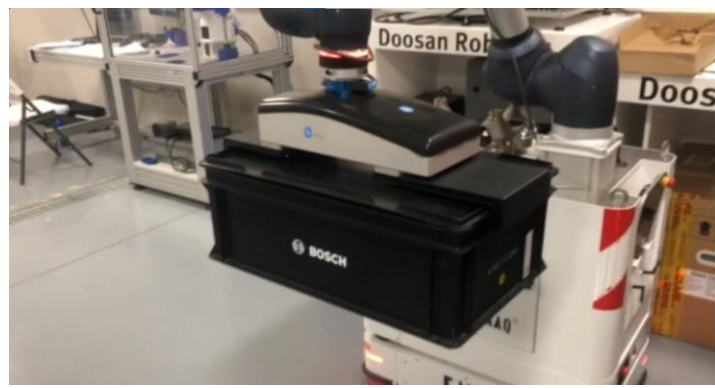
Figure 52 - Static simulation of the boxes to be manipulated.

4.4.3 Fingers Test

The fingers were manufactured using 3D printing, and manipulation tests using a Doosan robot were conducted at Europneumaq facilities to validate the simulations. The tests demonstrated that the gripper could successfully grasp the two types of boxes (Figure 53).



(a)



(b)

Figure 53 - (a) Grip test of the 400x300 box; (b) Grip test of the 600x400 box.

In this project, the gripper mounted with the designed fingers has a total weight of 4.7kg. Although, at first glance, it appears that the arm can transport the boxes since the total weight to be handled is 12.7 kg, this doesn't necessarily mean that the arm can manipulate the maximum weight of the *Targets*. As previously mentioned, the height of the gripper must not be too large because it would affect the arm's desired performance. This limitation happens because the payload of the arm varies depending on the distance from the manipulator's mounting flange to the center of gravity of the object being manipulated, as depicted in Figure 54.

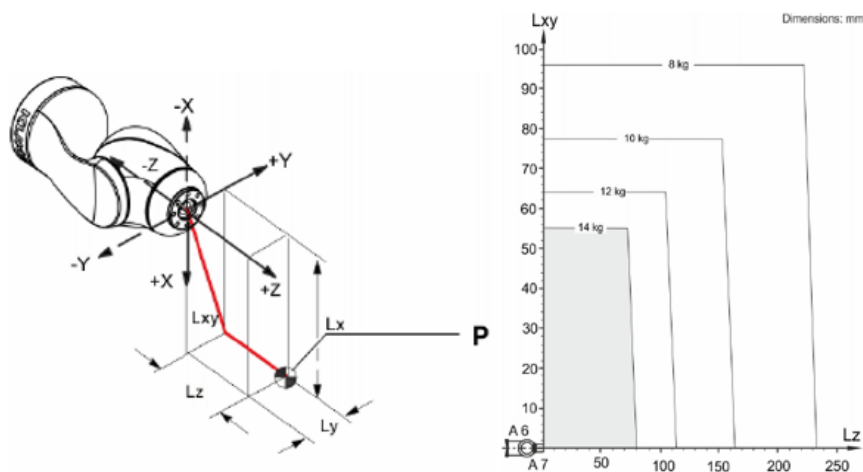


Figure 54 - Payload diagram of KUKA LBR iiwa 14 R820 manipulator. Adapted from (KUKA, 2019).

Due to the significant difference in weight between the gripper and the box, it can be inferred that the center of gravity of this system would be located somewhere within the

Target. By analyzing the payload diagram, it is evident that for the manipulator to work with the Gripper+*Target* system, the value of L_z must be less than approximately 100 mm. However, the gripper height alone already exceeds this value, as shown in Figure 44. Therefore, it is concluded that even though the gripper meets the initial project requirements, its considerable distance from the flange to its base inevitably reduces the payload capacity of the arm.

4.5 Maximum Number of *Targets* to be Manipulated

Since the Kuka LBR iiwa 14 R820 robotic arm is mounted on top of the mobile platform and given the fact that a dolly at the Bosch factory floor can support more than one *Target*, it is important to determine the maximum number of *Targets* the robotic arm can reach. An analysis was conducted for the mobile platform in different positions, concluding that the maximum number of *Targets* that the robotic arm can grasp from the dolly is four (Figure 55). Beyond four boxes, the arm cannot grasp any additional *Targets* from above.

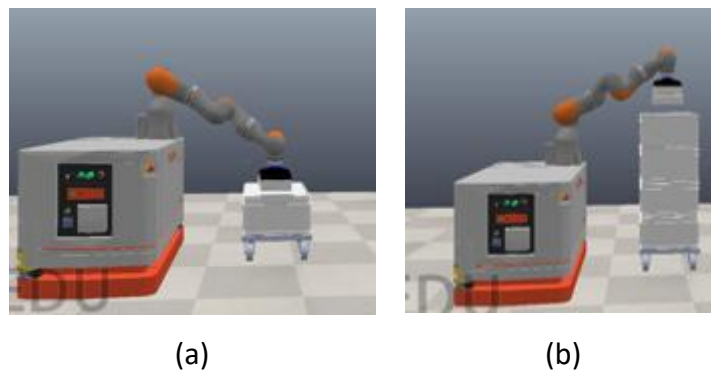


Figure 55 – (a) Manipulator grasping the first box from the dolly; (b) Maximum number of *Targets* placed on a dolly that the arm can reach.

Another crucial aspect to consider in the study is the maximum number of *Targets* the mobile platform can transport to the workstations. This number depends on the robotic arm's ability to place the boxes along the platform. Since the *Targets* to be handled can have two different dimensions, the analysis was carried out for three distinct cases: the maximum number of 600x400 boxes placed on the base of the mobile platform, the maximum number of 400x300 boxes that can be placed on the base of the platform, and the maximum number of *Targets* comprising both types of dimensions. For the first case, it was determined that two

600x400 boxes could not fully fit on the base of the KMR. Additionally, the arm can only place two stacked boxes due to its configuration, which does not allow for the addition of a third box (Figure 56).

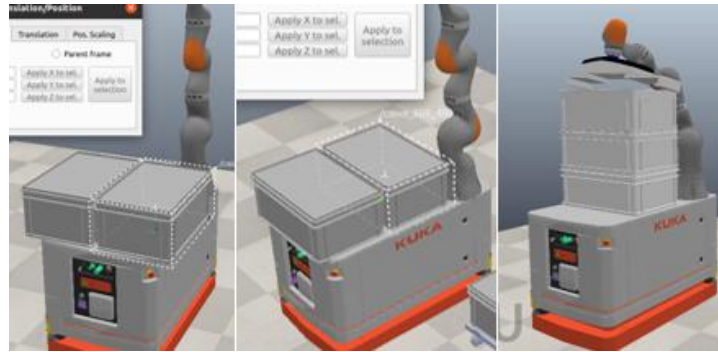


Figure 56 – Study of the maximum number of 600x400 boxes on the platform.

Regarding the 400x300 boxes, various distributions of these boxes at the platform's base were examined to determine which would allow for the maximum number of boxes. It was observed that the KMR could transport 4 boxes (Figure 57).



Figure 57 - Study of the maximum number of 400x300 boxes on the platform.

In the last case, different distributions were also considered for the two types of boxes. It was concluded that only two 400x300 boxes and two 600x400 boxes could be transported (Figure 58).



Figure 58 - Study of the maximum number of boxes of both types on the platform.

Although the number of boxes to be transported is low, Bosch requires the platform to be able to carry at least three small boxes and three large boxes. As such, a structure will be developed to accommodate this number of boxes or more mounted on the KMR. A conceptual design of this structure is shown in Figure 59. The 400x300 boxes would be positioned on the top of the platform, whose placement is inspired by current solutions as presented in (KUKA AG, 2024). Meanwhile, the 600x400 boxes would be stacked at the back of the KMR.

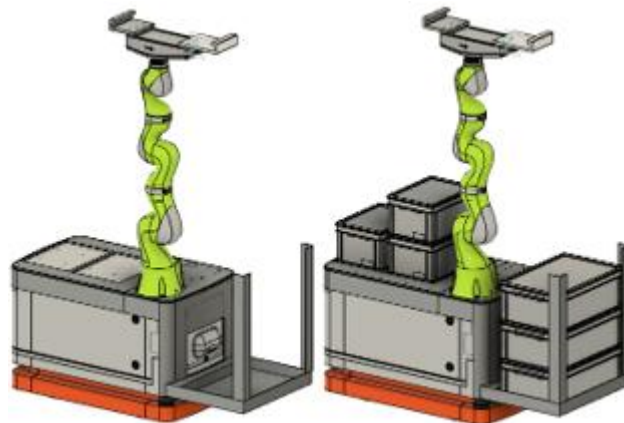


Figure 59 - Conceptual Structure to transport the boxes at the KMR.

Even though the structure presented in Figure 59 is merely a conceptual design and not the final version to be implemented in the project, certain aspects must be considered in the development of a similar solution. One critical factor is the minimum height of the framework supporting the larger boxes. This rear structure must not interfere with the KMR's LiDAR sensors, therefore, its height must be at least 180 mm, as shown in Figure 60.

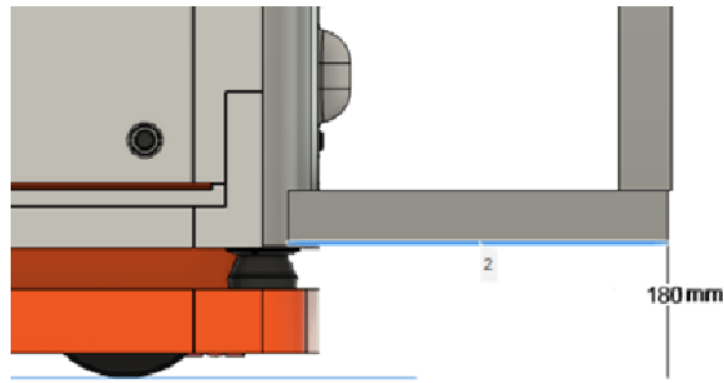


Figure 60 - Minimum height required for the rear framework that supports the larger boxes.

Moreover, the fittings designed to hold the boxes must provide sufficient clearance to secure them in place while preventing movement during the platform's displacement. In the presented conceptual structure, the same clearance used in the access ramps was considered. Specifically, a clearance of 10 mm was applied to each box's retention slots dimensions. Figure 61 illustrates the dimensions designed for each box type.

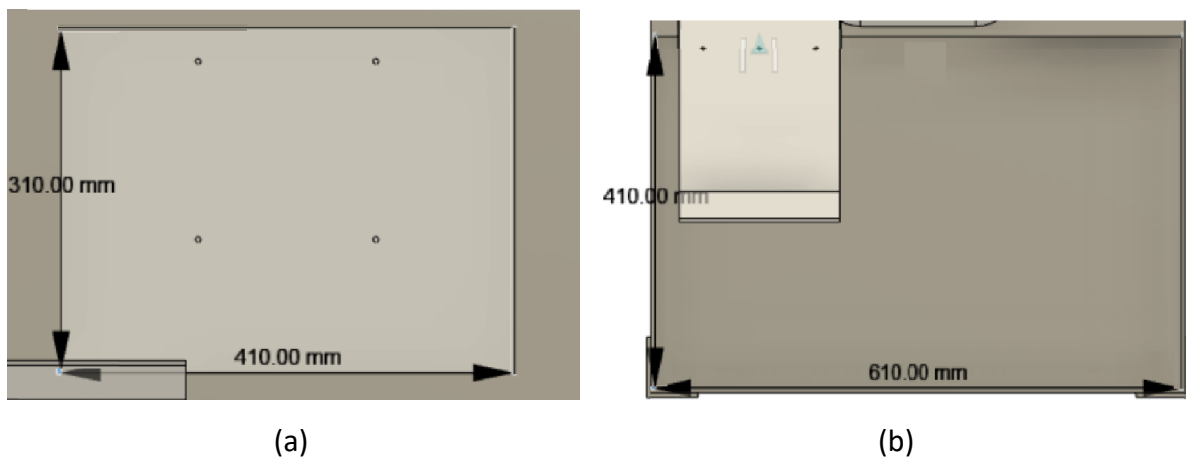


Figure 61 – (a) Top view of the retention slot at the KMR base for securing small boxes; (b) Top view of the retention slot in the rear structure of the KMR for securing the stacked large boxes.

Since a structure similar to the one presented has not yet been developed, this work used the presented design for further trajectory planning.

Using the proposed conceptual structure, an analysis was conducted to determine the maximum number of boxes the platform could transport beyond the desired six. It was observed that the arm could transport four boxes positioned at the base of the KMR (Figure 62 – (a)). However, for the arm to reach the fourth box (Figure 62 – (b)) and place it on the

access ramps, it is necessary to first remove the box closest to the manipulator to avoid collisions along the trajectory.

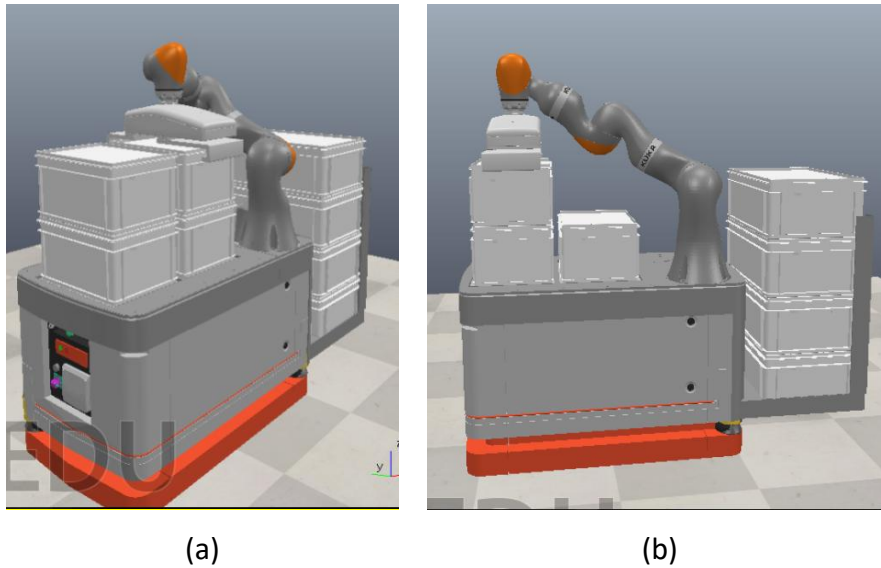


Figure 62 - (a) Maximum number of 400x300 boxes positioned at the platform's base; (b) Manipulator reaching the fourth 400x300 box.

The arm can also pick up four large boxes at the platform's rear (Figure 63).

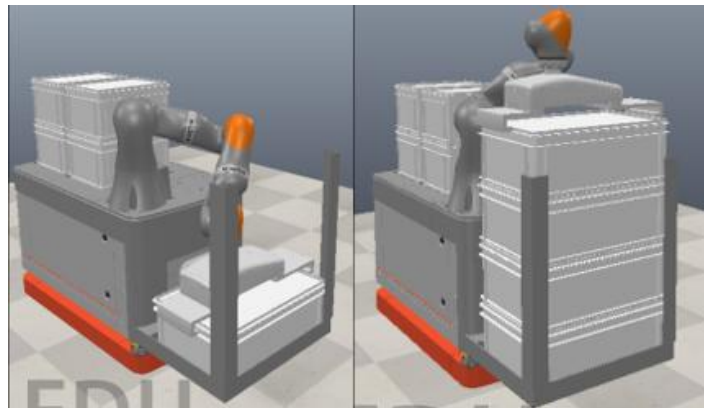


Figure 63 - Study of the maximum number of 600x400 boxes at the rear of the platform's structure.

4.6 Configuration in MoveIt

To execute the robot's motion in CoppeliaSim, it is necessary first to configure the robot in MoveIt to enable planning. For this purpose, the LBR-Stack directory, a collection of packages of the KUKA LBR iiwa 14 R820 manipulator already compatible with ROS2 humble,

was used. The `lbr_fri_ros2_stack` folder (mhubii, 2021) includes the robot description package `lbr_description` that has the URDF of the robot and its meshes, a MoveIt configuration package for simulating the robot in parallel with Gazebo, and packages that provide communication with the real robot through the hardware plugin `lbr_ros2_control::SystemInterface` (mhubii, 2024). To integrate the manipulator with the attached gripper in the RViz interface, it is necessary first to create an executable ROS 2 description package for the 2FGP20 gripper. This was accomplished using an Add-In script (fusion2urdf ROS2) from (Dheenadhayalan, 2021) for Autodesk Fusion. The created package (Figure 64) provides a URDF folder containing the URDF file and a Gazebo file for the gripper, a mesh folder containing the STL files of the gripper components referenced in the URDF file, and a launch folder with launch files to initialize the gripper in Rviz and Gazebo.

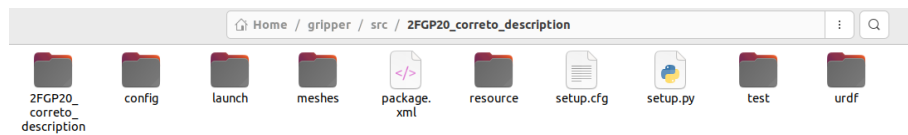


Figure 64 - ROS2 Gripper package generated from Autodesk Fusion.

After building this package, it was necessary to establish the binding between the arm and gripper packages in MoveIt. To accomplish this, the `xacro` file in the `lbr_description` package, which is responsible for reading the URDF of the KUKA manipulator, was modified to include the URDF from the gripper package. The base of the gripper was attached to the tip of the KUKA arm by settling the base link as a child to the manipulator's end-effector link. After connecting the gripper to the arm, the `moveit_config` package, as the one shown in Figure 28, needs to be re-built (or overwritten in this case) because it is essential to update the planning groups, collision checking, and controllers to include the respective gripper. Thus, the MoveIt Setup Assistant, described in Chapter 3.4, was used to configure the arm and gripper setup. Figure 65 illustrates some of the many steps involved in configuring the robotic arm, such as defining the hand planning group and the MoveIt controller named `hand_controller`, which has associated the prismatic joint of the gripper: `Deslizador2`, specifying fixed poses as the home position for the robot, and the hardware controller to control the gripper joint, which is of the type `position_controllers/GripperActionController`.

which contains 24 planners from the OMPL library, and the optimization-based planner CHOMP. The selection of these pipelines is based on their popularity in MoveIt, where the OMPL planners will be used for generating quick, time-efficient path plans suitable for real-time applications at Bosch Car Multimédia BRgP installations, and the CHOMP will be employed when it is necessary to produce smooth and legible trajectories. The launch file reads the planner from the `ompl_planning.yaml` and the `chomp_planning.yaml`, files. These files aren't automatically generated by the MoveIt Setup Assistant and need to be created manually within the MoveIt configuration package. As explained earlier, several nodes, like `move_group` and `rviz_node`, need to be launched, which start the Rviz2, `static_tf`, and `robot_state_publisher`. The arm and gripper controllers also need to be launched to ensure the movement of the robot's joints and the gripper in Rviz.

It's important to remember that since the goal of this work is to run the RViz simulation with CoppeliaSim, the hardware plugin used is the `mock_components/GenericSystem`. An important note to have in consideration is that when the manipulator is picking the boxes from the Dolly Supply and placing them in the access ramps, the boxes are full and have loose lids. Thus, the manipulator can't rotate the end-effector too much after grasping the boxes to avoid shaking the electronic components inside, preventing damage, or even dropping them while executing the planned path. So, the robot's movements must be constrained during the pick-and-place operations. MoveIt is constituted by OMPL's Constrained planning capabilities for this purpose. The OMPL-constrained planning requires the addition of configuration parameters in the `ompl_planning.yaml` file or, like in this work case, within the launch file, responsible for settling the robot to Rviz. The parameters are the `enforce_constrained_state_space`, which allows for planning within a constrained state space, and the parameter `projection_evaluator`, which helps discretize a high-dimensional state space by mapping it onto a lower-dimensional Euclidean space through projections (PickNik Robotics, 2024f). The visualization of the manipulator in Rviz is illustrated in Figure 66.

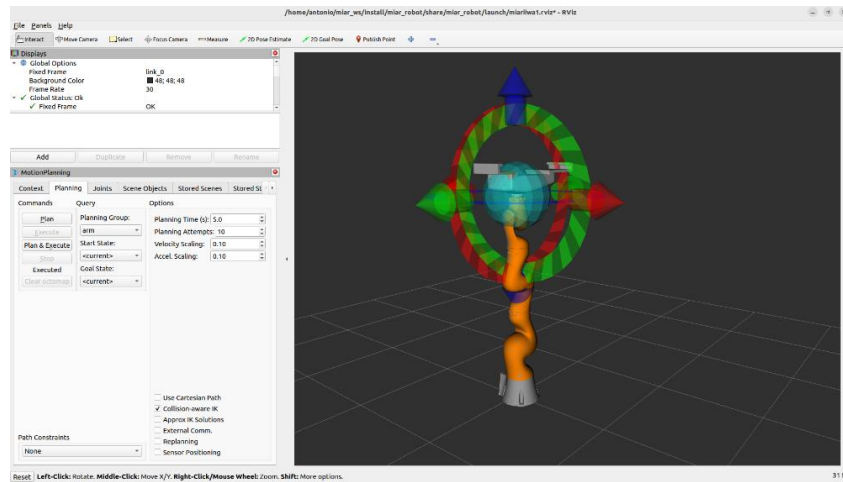


Figure 66 - KUKA LBR iiwa 14 R820 with 2FGP20 gripper in Rviz.

4.6.2 MoveIt-CoppeliaSim Communication

In MoveIt, when a robot is executing a trajectory to a specific point, its joints, including the gripper joint, are always read by the `/joint_states` topic. To avoid trajectory errors in CoppeliaSim, it is essential for the robot to replicate the same joint movements as in MoveIt without any delay. Hence, ROS2 communication was made between the two software. The communication logic followed the diagram in Figure 67, where CoppeliaSim sends the poses of multiple objects in the manipulator's workspace to MoveIt, while MoveIt continuously sends the robot's joint values to CoppeliaSim.

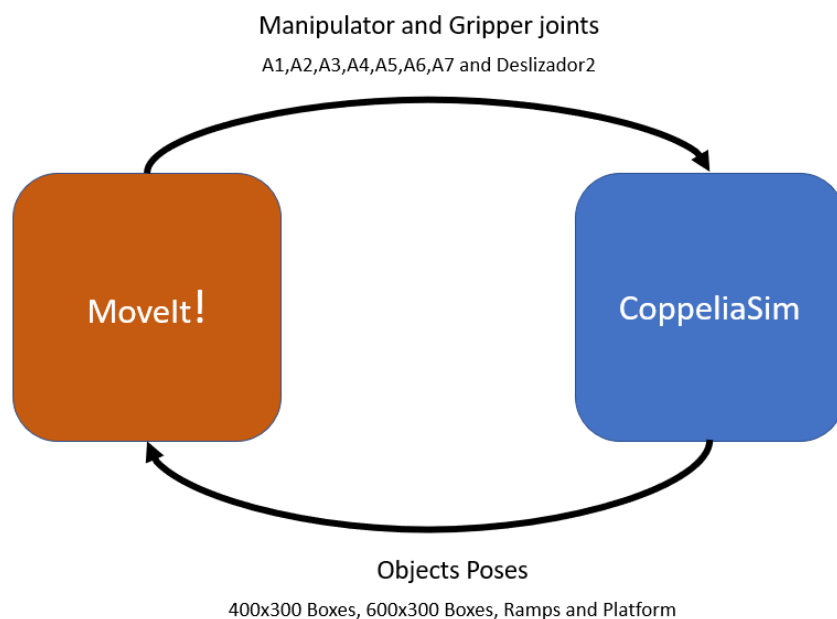


Figure 67 – MoveIt-CoppeliaSim Communication Diagram.

A node named `/joint_state_subscriber` was created to subscribe to the `/joint_states` topic, reading the robot's joint values that are provided in the `sensor_msgs/msg/JointState` message, which contains data describing the state of the controlled joints in MoveIt, including the joint names, positions, and velocities (ros2.org, 2020). This node publishes the robot joints to a new topic whose message type is readable in CoppeliaSim. For this purpose, the message type: `std_msgs::msg::String` was used. In the CoppeliaSim MIAR project scenario, a child script associated with the robotic arm was made. This script subscribed to the newly created topic and applied the joint positions in real time to the CoppeliaSim manipulator.

To synchronize the opening and closing of the gripper joint in CoppeliaSim with its counterpart in MoveIt, a topic named `fechar_gripper` was created in the main code and was responsible for executing the gripper's finger movement. Each time the gripper is commanded to grasp or release an object in MoveIt, the gripper's joint position is set as the message to be published on this topic. In CoppeliaSim, the gripper's child script subscribes to this topic, reads the joint position, and applies it as the value for its joint.

To plan the robot's movement in MoveIt, the arm must be aware of the obstacles around it to detect them and provide collision-free trajectories. To solve this problem, every model in CoppeliaSim that impacts the manipulator's workspace received a child script. This script reads the models' poses relative to the arm's base frame and publishes them in a topic, with orientation given in quaternion format. To make these objects appear in the Rviz environment, a set of code was developed to read the poses from the topics associated with each CoppeliaSim model and collision objects were created by reading the models' stl files to generate meshes. The objects were then published to the `/collision_object` MoveIt topic, which is responsible for placing objects in Rviz. In Table 5, it is possible to visualize the message types along with the associated topics that facilitate communication between the two software.

Table 5 - Topics and messages used for the MoveIt-CoppeliaSim communication.

Topic	Message	Goal
/joint_states	sensor_msgs/msg/JointState	Used to know the states of the robot's joints.
/joint_state_string	std_msgs/msg/String	Publishes the robot's joint positions to CoppeliaSim.
/fechar_gripper	std_msgs/msg/String	Sends the gripper joint position in the grasp and release actions.
/caixa400_300_position_orientation	geometry_msgs/msg/Pose	Sends the pose of the objects located in the robot's workspace.
/caixa400_300_position_orientation_1		
/caixa400_300_position_orientation_2		
/caixa400_300_position_orientation_3		
/caixa600_400_position_orientation		
/caixa600_400_position_orientation_1		
/caixa600_400_position_orientation_2		
/caixa600_400_position_orientation_3		
/caraca_position_orientation		
/plataforma_position_orientation		
/rampa1nivel_410_position_orientation		
/rampa1nivel_610_position_orientation		
/collision_object	moveit_msgs/msg/CollisionObject	Creates collision objects in Rviz environment.

For a more detailed understanding, Appendix C provides a visualization of how the communication is established between the nodes and topics responsible for sending the manipulator's joints from MoveIt to CoppeliaSim. The nodes and topics involved in sending the poses of the objects in the CoppeliaSim simulation scenario to Rviz are presented in Appendix D. These connections were obtained using the ROS tool `rqt_graph`, where the nodes are represented by an oval format and the topics by a rectangle.

The difference in the poses generated by the two software in relation to the world frame can significantly compromise the arm's ability to manipulate objects correctly in both scenarios. Accuracy errors in data transmission exceeding 14 mm can prevent the arm from gripping the boxes properly. It is essential that both scenarios are identical. Thus, the URDF, used to visualize the manipulator with the gripper attached in Rviz, was utilized to create them

in the CoppeliaSim environment. This ensures no differences between the two scenarios (Figure 68).

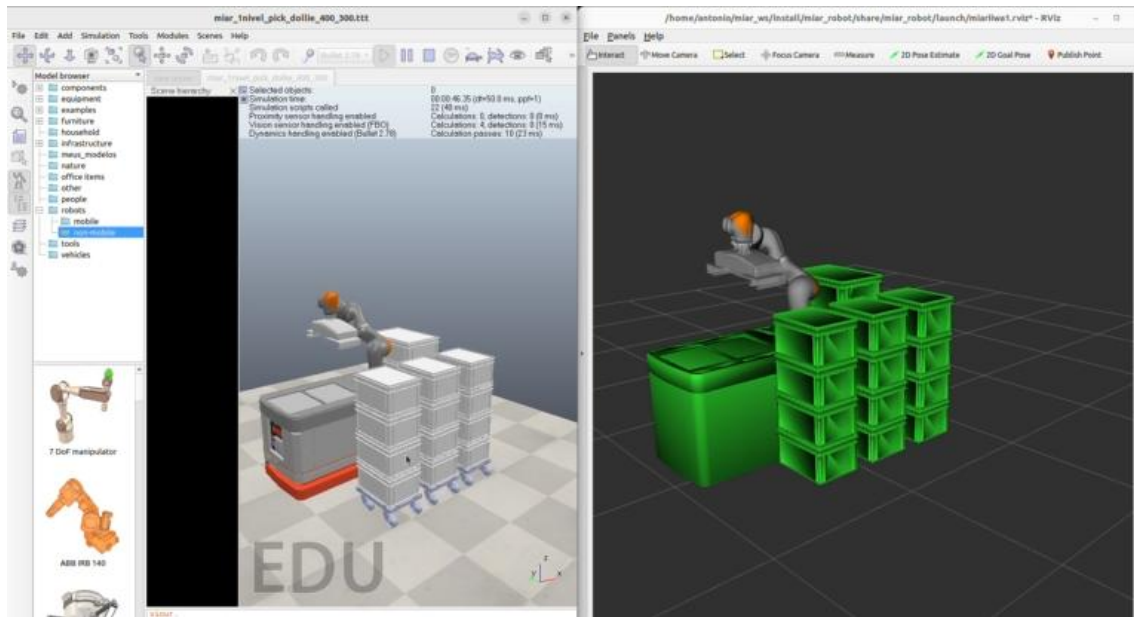


Figure 68 - CoppeliaSim-Movelt Communication.

4.7 Ramp Analysis and Optimization

An important aspect that needed to be studied before starting the pick-and-place operations was to confirm whether the “as is” access ramps would enable the manipulator to pick the boxes. A study was carried out to investigate this situation.

Concerning the two-level ramps, it was confirmed that this type of ramp is unsuitable for the pick and place tasks. For both the operations of picking empty boxes from the lower level and depositing full boxes on the middle level, the robotic arm did not have configurations that would enable it to perform these tasks. The dimensions of the set arm and gripper and slight height differences between access levels prevent the manipulator from approaching the ramp without colliding with it. Furthermore, due to its high height, the arm cannot reach or drop the full boxes on the upper level (Figure 69). This issue automatically limits the possibility of increasing the height between the levels.

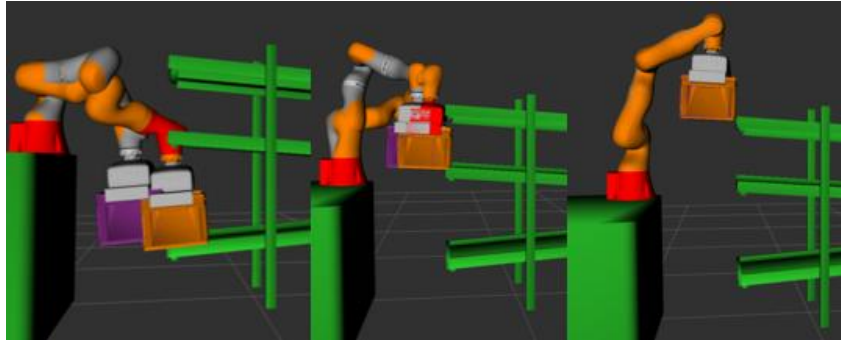


Figure 69 - Attempt to collect or place *Targets* by the manipulator on the different levels of the ramp.

Regarding the one-level ramps, with the KMR positioned at different distances from the 1200 mm corridor, it was concluded that the redundant manipulator could not pick up empty boxes from the lower level because the arm links collided with the upper level (Figure 70 – (a)). Several solutions were proposed to allow the arm to access this type of ramp and grab the boxes, such as modifying the height of the ramp levels. In an iterative process, one-level ramps with different height levels were evaluated based on the distance between the KMR and the ramps in the corridor. It was concluded that the lower level height should not be reduced. Doing so would not only cause ergonomic strain on the human operators on the opposite side of the ramp, forcing them to bend further to place the boxes, but would also compromise the path planning for the manipulator. Lowering the level would require the platform to move closer to the ramp so that the arm could reach the box, resulting in increased difficulty when removing the box due to the reduced distance between the platform and the ramp. For the upper level, an increase in the height of 200 mm was determined to prevent the robot from colliding with the ramp. However, the height could not exceed 300 mm, regardless of the platform's distance from the ramp. This proposal, however, was deemed unsuitable because it would require that all motions to remove the box from the ramp be performed using inverse kinematics (IK) instead of a global planner due to the limited manipulation space that would significantly increase the time required for the planner to find a feasible solution. This strategy also would force the robot to remove the boxes from the lower level by the side of the ramp, as there would not be enough clearance to remove the box directly between the platform and the ramp. Another suggestion was to design the ramp levels as a staircase (Figure 70 – (b)). This design would eliminate the issue of the arm colliding with the upper level when picking a box from the lower level. However, it would reduce the storage capacity of the upper

level and once again create ergonomic disadvantages for the human operators, as the upper level would need to be lowered for the arm to reach it.

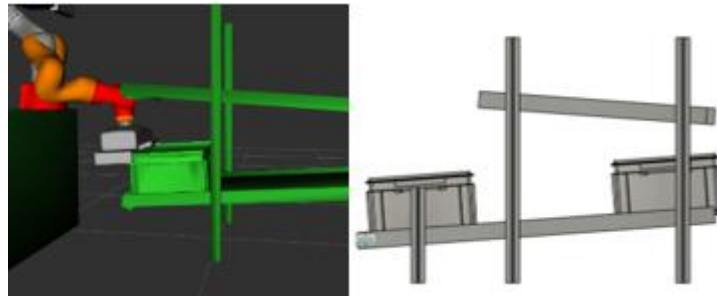


Figure 70 - (a) Collision of the manipulator's links with the ramp upper level; (b) Proposed staircase ramp.

It was suggested that the front bar of the upper level be cut to facilitate the entry of the robotic arm into the ramp for reaching the lower level. At the front part of the upper level, only the roller rails located at the corners would remain to support the boxes placed at the level's entrance. The support structure that holds all the roller rails and connects them to the rest of the ramp structure would be repositioned further back not to hinder the arm's entry into the ramp. A study was carried out to define the optimal entrance cut for the ramps, ensuring their suitability for the project. From this study, it was concluded that the ideal ramp for the 600x400 boxes should have an entrance cut of 500x350 mm, while the ramp for the 400x300 boxes should have an entrance cut of 300x350 mm. Besides, both ramps would require the levels to be raised by 20 cm (Figure 71).



Figure 71 - "To be" one-level ramp.

With the lower level raised by 20 cm, it would be possible to perform arm movements to pick up the box without resorting to numerous intermediate inverse kinematic (IK) movements. Another advantage is that the platform could be positioned farther away from the ramp, enabling direct removal of 600x400 boxes from the lower level. The increased height of the lower level in the pick operation would facilitate the placement planning of the boxes in the platform, as the lower level of the ramp would be closer to the platform's base level (Figure 72). For picking operations, the platform should be positioned 455 mm away from the ramps that store the 600x400 boxes and 435 mm from the ramps that hold the smaller boxes. For placing operations, the platform must be set at a distance of 185 mm from the ramps. The technical drawing of the new type of ramp can be seen in Appendix E.

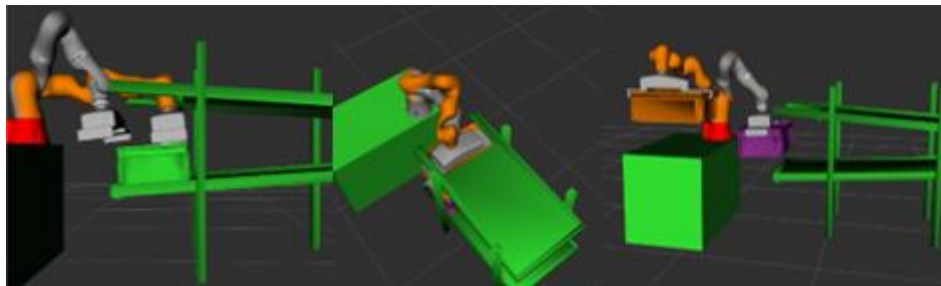


Figure 72 - Picking the 600x400 box on the optimized ramp.

A study was also conducted to evaluate whether the proposed modifications for the one-level ramps could render the two-level ramps suitable for future applications. In this scenario, the upper and middle levels were designed with a cut of 500x350 mm for the 600x400 boxes and a 300x350 mm cut for the 400x300 boxes. Using the original "as is" level heights, it was observed that the arm still collided with the levels during the placement tasks (Figure 73 – (a)). However, it could now reach the empty boxes on the lower level (Figure 73 – (b)).

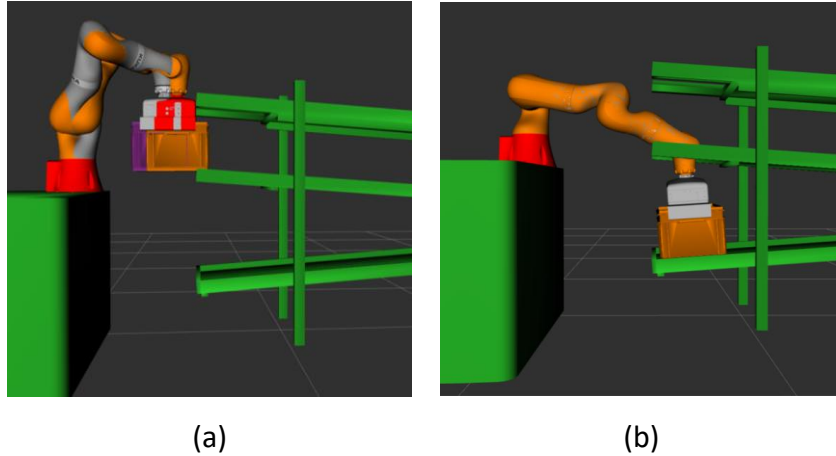


Figure 73 - (a) Collision between the robotic arm and the two-level ramp with entrance cuts; (b) Robotic arm retrieving an empty box from the two-level ramp with entrance cuts.

Further analysis was conducted to investigate how adjusting the heights of the ramp levels could improve their suitability for manipulation operations. From this study, it was concluded that all levels needed to be lowered to enable the robot to place boxes on the upper and middle levels without collisions (Figure 74).

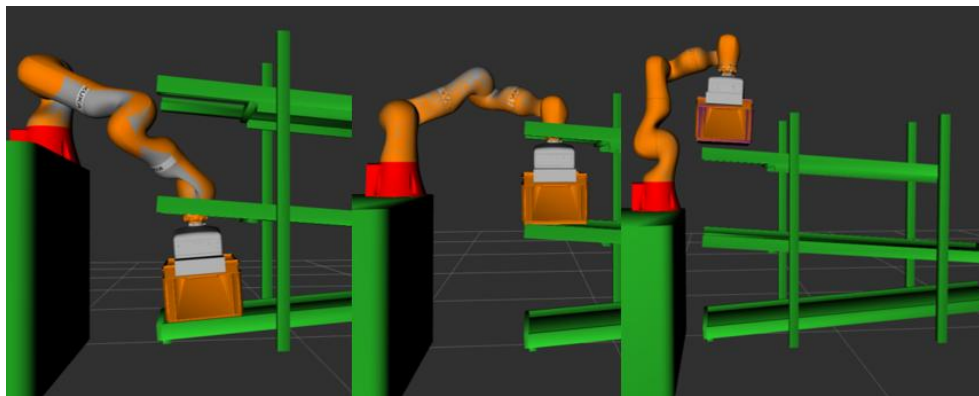


Figure 74 - Box placement and retrieval across different levels of the optimized two-level ramp.

An example of an optimized two-level ramp that supports pick-and-place operations is presented in Figure 75.

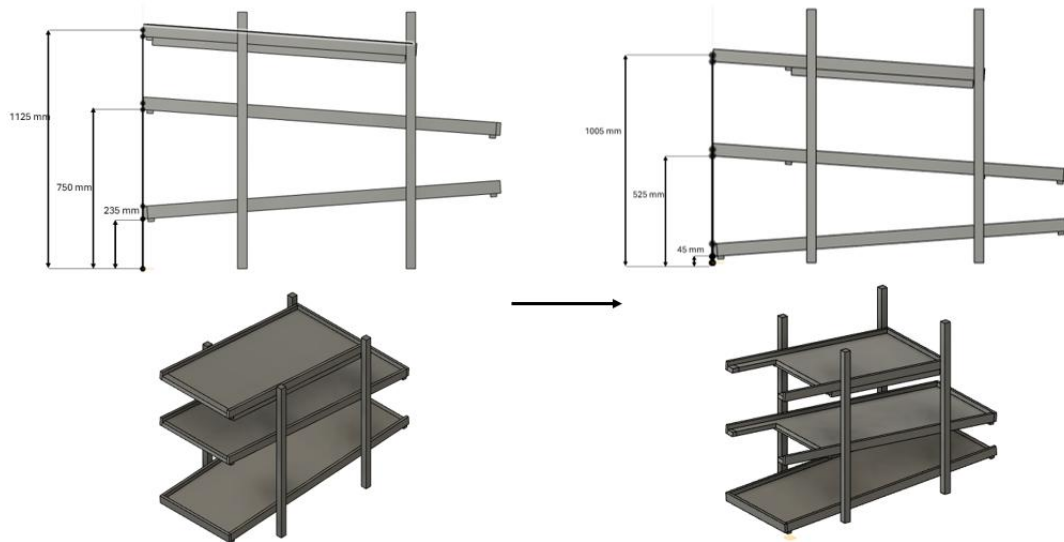


Figure 75 - Example of an optimized two-level ramp.

Although the arm can reach all levels of the two-level ramps, these ramps were not considered for the pick-and-place operations studied in this work due to issues of spatial clutter and the necessity of reducing the height of all levels. Such height adjustments would impose ergonomic strain on human operators, particularly when placing boxes on the lower level. Moreover, for the arm to access the lower level, it must be stretched, requiring the platform to be positioned closer to the ramp. This is problematic for larger boxes, as the limited space between the platform and the ramp would necessitate repositioning the platform further away from the ramp to remove the box. Thus, executing a single manipulation operation would require additional platform movements, complicating the overall process.

4.8 Pick-and-Place Operations in MoveIt/CoppeliaSim

In the MIAR project, there are eight manipulation scenarios for the pick-and-place operations, four of which relate to picking and placing the two types of *Targets* on the access ramps and four of which relate to picking and placing the respective *Targets* on the dollies. So, eight operation codes must be developed, each adapted to the conditions of its specific scenario.

The correct execution of trajectories in the manipulator for the MIAR project was managed using only two launch files. The first launch file is responsible for creating the Coppelia scenario for MoveIt, and the second is responsible for executing the desired task.

Regarding the first launch file, it is composed of a set of nodes that are initiated sequentially. Each node represents a process and is defined by its executable, the package it belongs to, and optional parameters such as the node name and output. Event handlers were used to register specific transition events between nodes to ensure the sequential execution of nodes (Open Robotics, 2024c). This is achieved using the RegisterEventHandler action along with the OnProcessExit event handler. The OnProcessExit is triggered when a specified node finishes execution, allowing the dependent node to start only after its predecessor has finished. This structured method ensures that all processes are executed in the correct order, respecting dependencies and preventing potential conflicts during execution. The first nodes in this sequence publish each model present in the physical simulator scenario as collision objects in Rviz. Depending on the number of models present in CoppeliaSim, the Rviz environment is adapted to match the simulation. Nothing is published as a collision object if the robot has no obstacles in the scenario. After the sequential run of the nodes responsible for publishing the poses, the final node settles the communication between the manipulator in MoveIt and the one in CoppeliaSim, as previously explained.

As already exclaimed, the second launch file handles the execution of the pick-and-place operations. Depending on the manipulation scenario selected by the user, one of the eight possible operations will be executed. So, this launch file depends on the desired operation and takes the name of the CoppeliaSim model the arm must pick or place as input. Consequently, this launch file is composed of a sequence of two nodes, both of which receive the launch file's input as an argument. The first one corresponds to a node that reads the pose of the CoppeliaSim model to be manipulated, provided as an argument, and publishes it in Rviz. The second node executes the planning code for the specified object. By structuring the process in this way, the scenario is always read before proceeding with the manipulation operation. If there is any positional variation in the models in CoppeliaSim, MoveIt will update this change, and the subsequent planning will take this into account. Following this logic, these codes could be easily adapted to real-world applications. In the first launch file, cameras would detect the real-time positions of the objects relative to the arm depending on where the platform stops near the ramps or dollies. In the second launch file, the camera would identify

the box that the arm needs to manipulate, and its pose would be sent to MoveIt, allowing the arm to execute the trajectories based on its workspace. This approach would be beneficial in situations where accuracy errors occur when positioning the KMR in each scenario. To provide a clearer understanding of the described method, Figure 76 presents a diagram outlining the procedure required to establish the execution of trajectories, where the launch files are represented in an oval format and the nodes responsible for publishing the objects in Rviz and for the pick-and-place algorithm by a rectangle.

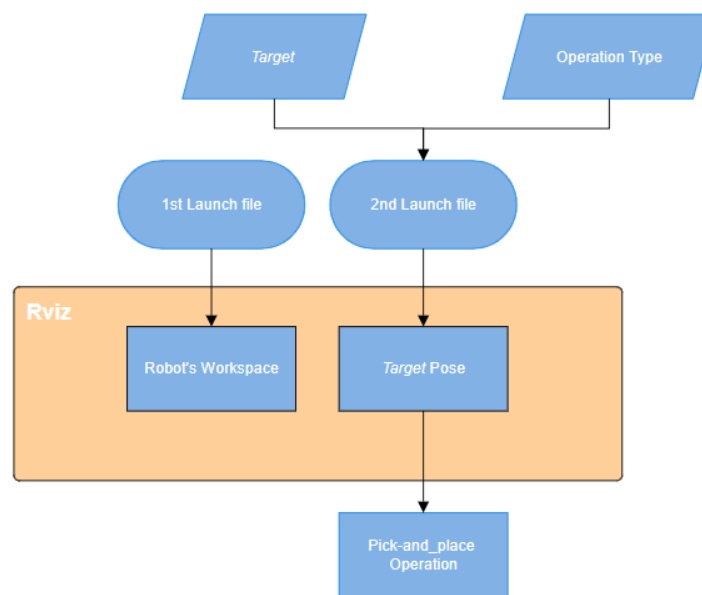


Figure 76 - Diagram of the procedure for initializing the Pick-and-Place Operations.

The pick-and-place codes were made using nodes that rely on the `move_group_interface` class, which provides easy functionality to communicate with the `MoveGroup` node. This class is particularly useful for motion planning, setting poses, joint configuration goals, and attaching or detaching added objects in the workspace (PickNik Robotics, 2024b). The planning groups needed to be declared to set up the `MoveGroupInterface` in each created node. The `move_group_arm` interface allows the planning and control of the manipulator's movements by interacting with the planning group arm, while the `move_group_hand` interface interacts with the planning group hand, which controls the gripper joint. The `PlanningSceneInterface` class was utilized to access and manipulate the planning scene, allowing objects to be added and removed from the environment. It was mainly used to consult the current state of the manipulator between

transitions in motion trajectories to ensure that the robot plans safe and known movements. Moreover, the `PlanningSceneInterface` is used at the beginning of each node to verify whether the object name provided as input to the launch file already exists in the Rviz environment. In case the object is not found, the node shuts down, causing the launch file to terminate. If the object is in the manipulation space, its pose is read, and all motion planning within the node depends on this pose. The planner used to control the arm is defined before initiating motion planning through the `move_group_arm` interface. A home joint configuration for the manipulator was also established to ensure uniformity across all required scenarios. For the 400x300 boxes, the arm is positioned in a joint configuration that ensures the robot can reach all the boxes placed on the platform's base. The same logic is applied to the 600x400 boxes, where the arm is positioned in a configuration goal that ensures it can pick and place the boxes at the back of the KMR, as shown in Figure 63. The arm is able to reach all four large boxes through IK without colliding with the support structure and without requiring changes to the robot's configuration during the up and down movements.

Figure 77 presents a general flowchart to illustrate the workflow of the developed pick-and-place codes. This flowchart outlines the initial steps, such as detecting the object name in the Rviz workspace and the subsequent steps related to the predefined processes for either the Pick or Place operation.

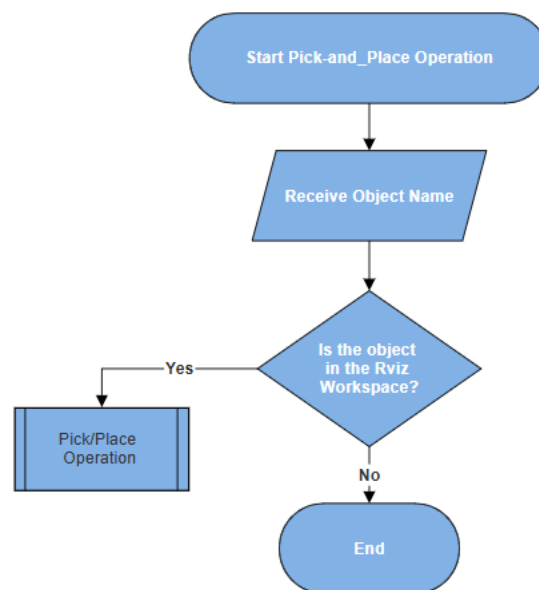


Figure 77 - General flowchart of the Pick-and-Place operation.

Below, there is a brief description of how each node works according to the desired operation.

The code used to pick both types of boxes from the access ramp follows a similar logic, differing only in the poses for placing the objects on the KMR. Before planning the manipulator's movements to pick the boxes from the lower level of the ramp, it is necessary to apply motion constraints to the robot. As often emphasized, the intention is to generate legible and smooth trajectories for the robot arm, ensuring that during execution, the manipulator's joint positions do not vary excessively or assume unforeseen values. For each operation, at the beginning of the planning in each node, a set of joint constraints is applied to guide the arm along the most predictable path to the goal and to prevent it from realizing unexpected movements during the trajectory. Furthermore, the orientation of the arm's end-effector must also be constrained because, as previously discussed, the boxes have loose lids and may be full, requiring the robot to handle them carefully during transport. In scenarios where the arm needs to manipulate empty boxes, a less restrictive tolerance is given to the orientation constraint compared to the full boxes. Therefore, in the pick scenario, a constrained path is planned to move the robot's end-effector to the ramp entry. Since the intention is to generate paths within the shortest possible time, each planning attempt is assigned a planning time of 1 second. It was considered that the planner is allowed a maximum of 10 attempts to find a solution. If the planner fails to compute a valid trajectory, the node shuts down, and the entire operation is canceled. For motion execution, the manipulator is restricted to a maximum velocity of 10% of its total capacity, and the same logic applies to the maximum allowable acceleration. Due to the limited space near the ramp, the arm is moved close to the box and subsequently to the grasp pose using the MoveGroup function `computeCartesianPath`. This function allows the robot to execute linear movements by IK following specified waypoints with a defined step size (I. Sucan, 2017). After grasping the box, the arm retreats from the ramp using a cartesian path and returns to the home position, once again, through the constrained planner. Finally, the arm normally approaches the place pose via planned motion and deposits the box using another cartesian path.

The flowchart presented in Figure 78 expresses all the steps related to the pick operation code. It is important to note that each planning process block is associated with a decision block that verifies the number of planning attempts and an execution block.

However, to avoid excessively extending the flowchart, these blocks are only explicitly shown in the first instance and subsequently represented by a simplified "Plan and Execute" block.

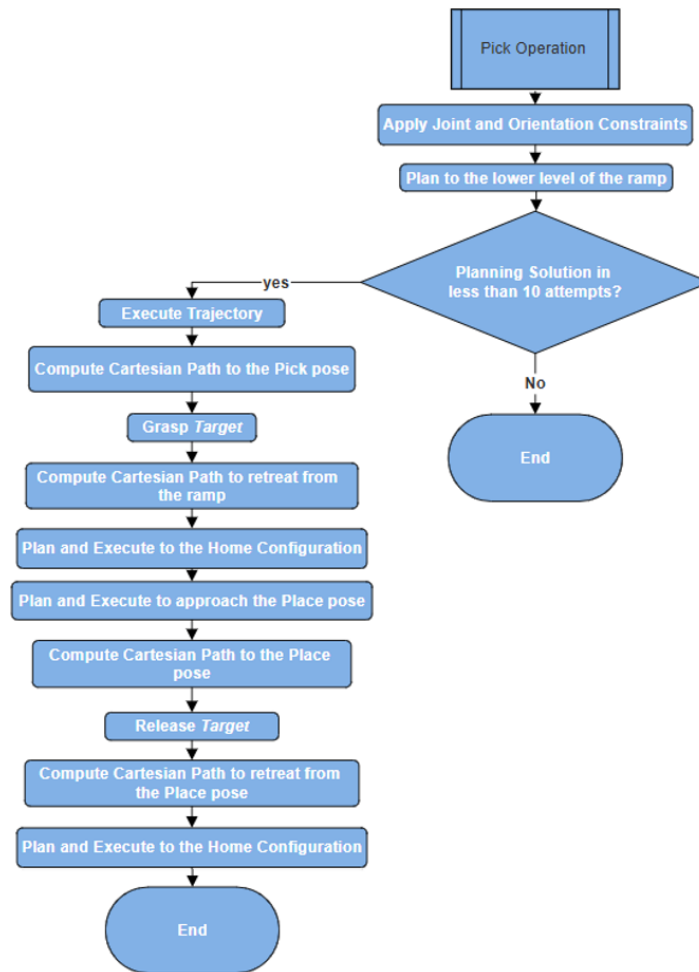


Figure 78 - Flowchart of the Pick operation.

To place the *Targets* on the ramp, the manipulator must retrieve the requested box from the KMR and place it on the upper level of the ramp. Before starting with the intended operation, the current joint positions are read and compared with the defined home configuration. If the arm is not in the desired goal, it is moved to the initial configuration before starting the task of placing the boxes. First, only joint position constraints are applied to pick the boxes from the platform, as the arm is already in a suitable pose to grab them. If the orientation constraints were added, the computation time required to find a feasible path would unnecessarily increase. Once the arm grips the box to be manipulated, it retreats to the home configuration, where it starts planning a new path to place the box on the upper level. Since the arm is now handling a full box and the planning process for placing it is more

challenging, an orientation constraint is considered to avoid issues during manipulation. Like in the picking operation, the cartesian path is used to approach the box for grasping and used to move away from the box after releasing it. Figure 79 illustrates the flowchart of the place operation.

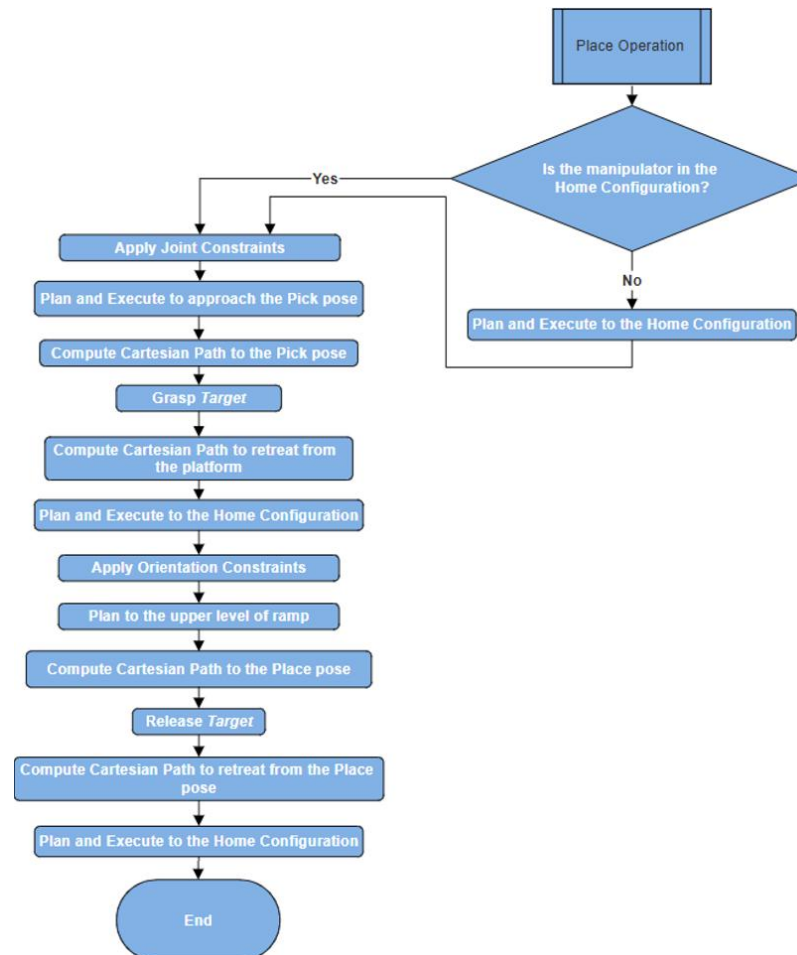


Figure 79 - Flowchart of the Place operation.

When it comes to picking and placing *Targets* in the Dolly Supply, the same principles are applied as in the previous cases. The flowcharts presented in Figure 78 and Figure 79 are equally applicable to these scenarios, with the primary difference being that, instead of a ramp, the obstacle to the manipulator's trajectories is a dolly.

5 Analysis of Results

This chapter details the tests and results obtained to validate the most suitable motion planner for the MIAR project.

Initially, an analysis is conducted comparing ten motion planners using their default settings for pick-and-place operations at the access ramps. Subsequently, the planner that demonstrated the best performance in the initial study is subjected to customization tests to assess its viability in generating smooth trajectories.

Finally, the chapter highlights the outcomes of the eight manipulation operations performed using the motion planner selected for the project.

5.1 Selection of the Ideal Motion Planner

As explained in Chapter 3, selecting the most suitable motion planner for the pick-and-place operations in this project is a key goal. The manipulator must perform its tasks in the shortest possible time, avoiding obstacles within its workspace and providing smooth and legible movements. Since there isn't an ideal motion planner for every manipulation scenario, the selection process was based on a comparison of different motion planners.

Several benchmarking studies have been conducted to identify the best planner solution for practical problems. Wesley Reis et al. (2019) used the MoveIt tool OMPL Planner Arena to compare various OMPL planners, such as Bi-TRRT, RRT*, and RRT. The evaluation metrics included total planning time and path length for a 6-DoF manipulator. From the effectuated analysis, the planners T-RRT and RRT-Connect yielded the best results. Gomes De Brito (2018) compared OMPL planners in MoveIt to choose the most qualified one for planning a vine pruning task. Five different tests were carried out, using maximum, minimum, and average planning time and the success rate of path planning attempts as evaluation metrics. Planners like BiEST, KPIECE, LazyPRM*, and RRT-Connect were highlighted. S. Yang et al. (2023) presented a benchmarking study of motion planners in cluttered environments. Having as metrics the meantime for performing feasible paths and the success rate of planning attempts for 12 different planners, RRT-Connect and BKPIECE were identified as standout options. S. Liu & Liu (2022) also made a benchmarking study, evaluating planners based on

success rate, path planning time, and a path smoothness metric to validate whether the trajectory is smooth. The latter metric measured the angle between three consecutive points, where values closer to 0 indicate smoother trajectories due to a higher rate of straight-line segments. This study concluded that the RRT-Connect planner is a good option for high-complexity scenarios.

Given this overview, it can be concluded that selecting the appropriate motion planner depends on the distribution of the manipulation scenario. Consequently, a motion planning benchmarking study was conducted for the MIAR project. The experiments were carried out on a personal computer using Ubuntu 22.04. The computer consisted of an MSI Katana 15 B13VGK-2040 laptop equipped with a 13th-generation Intel Core i7 processor, 32 GB of RAM, a 1 TB solid-state drive (SSD), and an NVIDIA GeForce RTX 4070 GPU. It is noteworthy that the comparison was exclusively conducted using MoveIt, employing the pick-and-place codes outlined in section 4.8.

A group of planners was compared across four different scenarios corresponding to the pick-and-place operations of the two types of boxes on the access ramps. These scenarios were considered due to their critical importance to the overall project and their complexity related to the limited space available for placing the boxes on the ramps. To take benefit of the best planner, the worst-case scenario was considered for each planning, where the robot's workspace was fully occupied by the maximum possible number of objects. For example, when the manipulator needed to pick up or place 600x400 boxes on the ramp, it was assumed that the platform was already full of 400x300 boxes, and vice versa.

The comparison involved the planners RRT-Connect, T-RRT, BiEST, KPIECE, and LazyPRM*, which were highlighted earlier in the reviewed studies. Optimized versions of some mentioned planners, such as PRM* and RRT*, were also included in the benchmarking. Additionally, planners like Bi-TRRT, RRT, and ProjEST were considered after satisfactory results were obtained during preliminary testing in the indicated scenarios.

When it comes to choosing the benchmarking metrics, the analysis considered total planning time, total execution time, path length, and the success rate in finding a solution to measure the performance of each planner. A smoothness factor was included to evaluate the quality of the generated paths. This parameter is derived from the metric used in the benchmarking study by Cohen et al. (2012) to measure path smoothness. For a path composed of n segments, the angle α_i between two consecutive segments is calculated, and the

following Equation 5.1 is applied. The lower the value of this metric, the smoother the trajectory is considered.

$$K = \frac{1}{n} \sum_{i=2}^n \alpha_i^2 \quad 5.1$$

In Movelt, each planner includes multiple parameters that can significantly influence the performance of the sampling-based planners. Since modifying and testing each parameter individually by planner would be a time-consuming process, all the mentioned planners were compared using the default parameters provided by the Movelt 2 package. A planning time of 1 second was specified for each planner, as the goal is to achieve fast planning, along with the constraints and motion kinematics explained in Section 4.8. Each planner was simulated ten times for each operation. The optimization-based planner CHOMP was initially considered for this benchmarking study. However, this planner typically generates long processing times, as highlighted in (S. Liu & Liu, 2022). Given the defined constraints and the settled planning time, CHOMP was unable to find a solution without adjustments to its parameters. As a result, this planner was excluded from the benchmarking study.

The benchmarking process was inspired by the OMPL Planner Arena from Movelt, utilizing Tukey Boxplots to compare the performance of each planner across different metrics. Similar to the Planner Arena, the plot analyses in this study exclude data from failed attempts by the planners. This exclusion is crucial for correctly interpreting the studied plots because some planners may present data from the total number of planning attempts, while others might show a subset. To address this issue, as done in the Planner Arena, a table showing the success rate of each planner is presented alongside the plots (Moll et al., 2015).

As the goal is to identify a suitable planner for all the required operations on the Bosch shop floor, any planners that fail to compute a solution within the 10 defined attempts are automatically excluded from consideration.

The analysis of the obtained results is thoroughly addressed in the subsequent section.

5.2 Analysis of the Motion Planners Comparison

As explained in section 5.1, a benchmarking study of ten OMPL planners was performed for the scenarios involving the access ramps, using the planning and execution times, path length, a smoothness metric, and the success rate in finding a solution as evaluation metrics.

The results are presented below for the place operation involving the 600x400 boxes, which consists of placing the boxes on the upper level of the ramp. Due to the many metrics and planners included in the comparison, the analyses for the remaining operations are presented in Appendix F. In the operation of placing the 600x400 boxes, it was observed that all the planners provided relatively short planning times for both the placement and retreat paths (Figure 80). However, RRT-Connect and BiEST exhibited slightly higher values compared to the others.

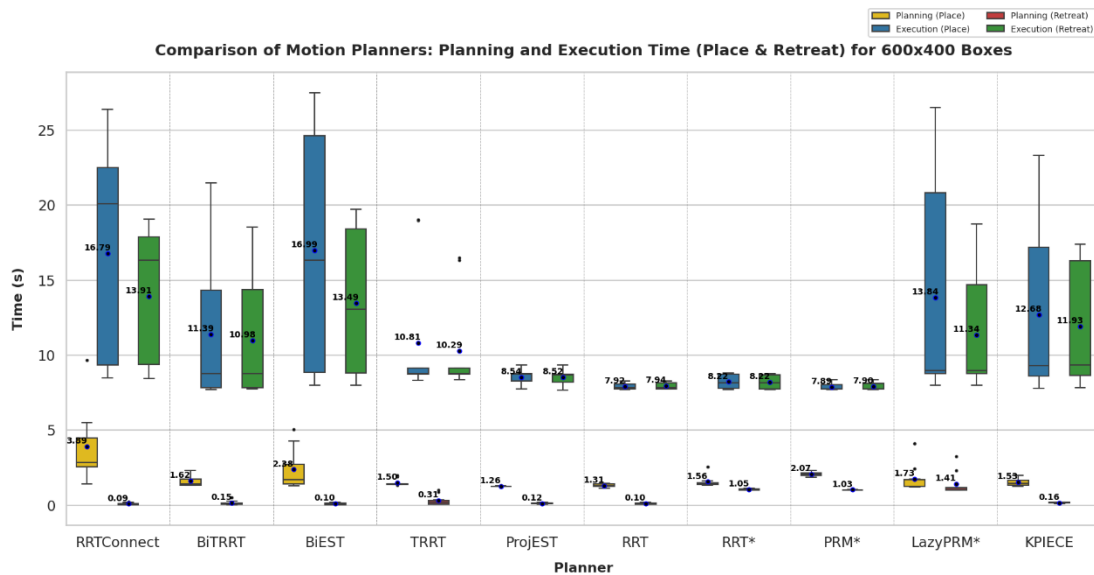


Figure 80 - Comparison of planning and execution times for the place operation of the 600x400 boxes.

A significant discrepancy was observed between the planners in terms of execution time. The planners that demonstrated the best results were ProjEST, RRT, RRT*, PRM*, and T-RRT. This latter planner contained outliers, one in the place execution time and two in the retreat execution time. These outliers resulted from execution times that exceeded the upper whisker, meaning values higher than the upper quartile (Q3) by 1.5 times the interquartile range. Although LazyPRM* and KPIECE had a high place and retreat execution times, it is

notable that half of their values are concentrated as their median values were below 10 seconds.

These execution time values are reflected in the path length of the entire operation generated by each planner (Figure 81). Planners with higher execution times, such as BiEST and RRT-Connect, produced longer paths, while planners like RRT, ProjEST, and PRM* yielded shorter paths.

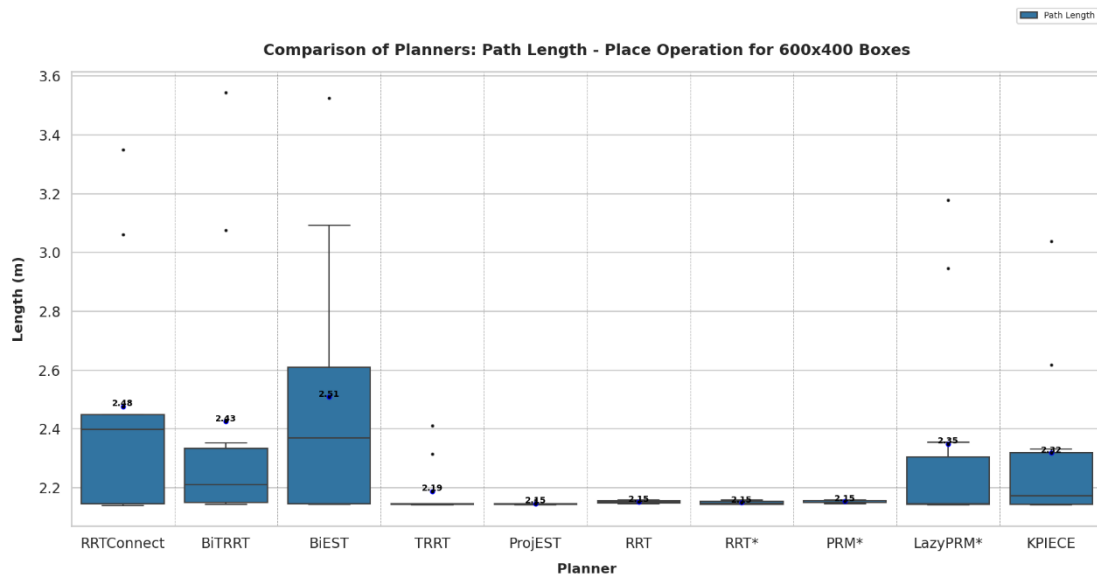


Figure 81 - Comparison of the path length for the place operation of the 600x400 boxes.

Using the metric explained earlier, smoothness was observed. Planners producing longer paths tended to create smoother trajectories (Figure 82). These planners prioritized the motion constraints specified in the operation code rather than finding the fastest path to the goal. As a result, they generated longer and unnecessary paths to yield more linear paths, while other planners allowed larger angles between the segments that composed the path.

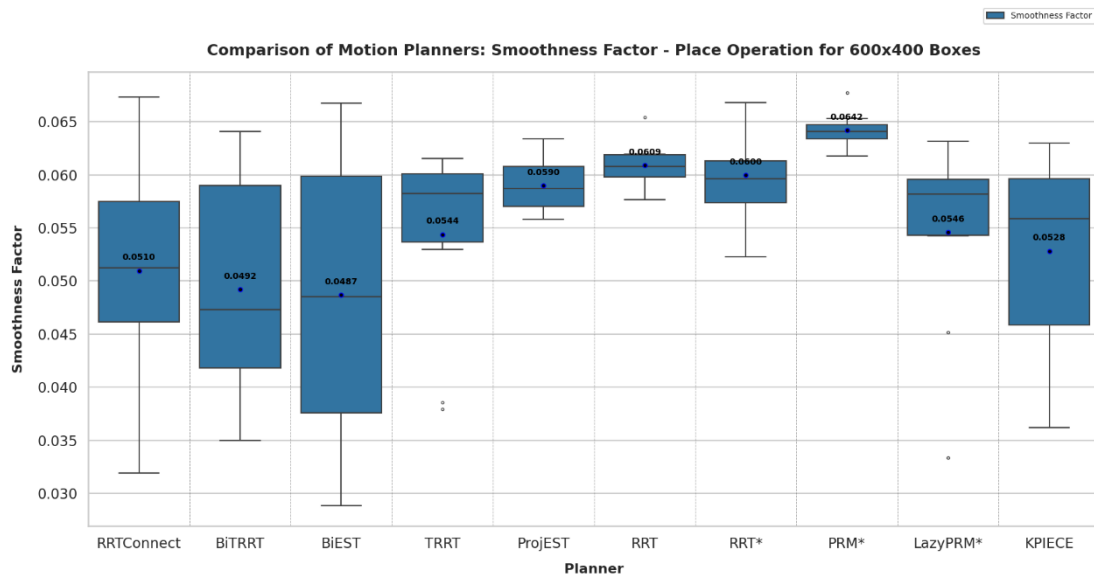


Figure 82 - Comparison of the smoothness factor for the place operation of the 600x400 boxes.

Regarding the success rate in finding a feasible solution, most planners successfully generated a valid path, except the RRT-Connect planner, which failed in one attempt. Only ProjEST, RRT, PRM*, and KPIECE consistently found a solution on the first attempt (Figure 83).

N° Attempts	RRT-Connect		Bi-TRRT		BiEST		T-RRT		ProjEST		RRT		RRT*		PRM*		LazyPRM*		KPIECE	
	place	retreat	place	retreat	place	retreat	place	retreat	place	retreat	place	retreat	place	retreat	place	retreat	place	retreat	place	retreat
1	2	9	10	8	6	10	10	9	10	10	10	10	9	10	10	10	8	8	10	10
2	3				2			1									1	1		
3	2			1	1												1	1		
[4-6]	1			1	1															
[7-9]	1																			
10																				
>10	1	1																		

Figure 83 - Number of attempts for the place operation of the 600x400 boxes.

From the pick operation of the 600x400 boxes presented in Appendix F, it is possible to analyze that only the Bi-TRRT and T-RRT planners achieved good results in terms of planning time and execution time for both the pick and retreat movements of the task, with Bi-TRRT showing outliers for the pick execution time. The remaining planners exhibited high execution time values for the pick movement, with very dispersed data. These results are also expressed in the path length metric, as these two planners generated the shortest paths.

Regarding the smoothness factor, the standout planners were once again Bi-TRRT, T-RRT, and RRT*. Unlike the previous operation, all planners successfully found a path in this operation. Bi-TRRT, T-RRT, PRM*, and LazyPRM* have achieved high success rates. Based on this analysis, it can be concluded that these two first planners were the best options for this operation.

For the place operation with the 400x300 boxes, all the planners generally presented good planning and execution times, with the ProjEST, Bi-TRRT, and RRT planners standing out. However, the RRT-Connect, LazyPRM*, and BiEST planners showed high execution times. The planners that created the shortest paths were Bi-TRRT, ProjEST, RRT, RRT*, and PRM*. Despite the favorable results presented by the Bi-TRRT planner, it is important to note that it could not find a feasible solution in all 10 attempts. This planner failed one time to find a solution for the placing operation and, consequently, for the retreating movement. RRT-Connect, BiEST, and LazyPRM* were the planners that contributed to smoother trajectories. However, once again, considering the success rate, the RRT-Connect and BiEST planners failed to find a solution at least twice. Therefore, the RRT-Connect, Bi-TRRT, and BiEST planners were excluded from the selection of the most appropriate planner for the MIAR project, regardless of their good performance in other operations.

In the pick operations with the 400x300 boxes, only two planners stood out in the generated times: Bi-TRRT and T-RRT. While Bi-TRRT demonstrated shorter planning times, T-RRT provided shorter execution times. Although this interpretation might suggest that T-RRT generated shorter paths, when analyzing the plot based on the operation length, it is verified that Bi-TRRT generated shorter solutions. This is confirmed by its lower mean, smaller interquartile range, and shorter whiskers. Despite being faster at generating solutions with minimal distances, both Bi-TRRT, T-RRT, and KPIECE displayed trajectories with less smoothness. This is confirmed by their higher smoothness factor values, which are approximately 0.05. In contrast, the RRT-Connect planner excelled in this metric. When examining the success rate, several planners, namely BiEST, ProjEST, RRT, RRT*, and PRM*, failed to find always a solution. Over 60% of the attempts executed by ProjEST, RRT, and RRT* were unsuccessful. Among all the planners, Bi-TRRT was the only one that consistently found a solution on the first attempt.

The conducted benchmarking study concluded that many planners could not produce paths for the operations involving the manipulation of the 400x300 boxes. This lower success rate compared to the tasks involving the 600x400 boxes can be attributed to the limited space between the platform and the access ramps. The smaller boxes are positioned at the base of the platform, making the required manipulator movements more challenging than picking or placing the larger boxes in the conceptual structure located at the back of the KMR. Among all the considered planners only three were able to find suitable solutions in all four scenarios,

which were KPIECE, LazyPRM* and T-RRT planners. Since T-RRT delivered the best overall results, performing well across all evaluation metrics, it was chosen as the planner for the MIAR project.

5.3 Optimization of the Motion Planner

T-RRT planner demonstrated satisfactory results for most box manipulation tasks. However, as presented in the previous analysis, its performance on the smoothness metric was the worst in the place operation for 400x300 boxes, and it also showed high values for the pick operation. One alternative to address this problem is further limiting the orientation constraint for the manipulator's end-effector. This adjustment would ensure safer handling of the filled 400x300 boxes and reduce the variability in trajectory generation. The T-RRT planner was customized by tweaking its parameters to achieve faster solutions without fully compromising path quality through an iterative process. The initial temperature, parameter that was mentioned in Section 3.2.4, was augmented as it is recommended that it should be high at the beginning of the algorithm to enable a broader exploration (Kavraki Lab, 2024). The `max_states_failed` parameter was reduced to accelerate the temperature increase after a few failed transitions to new states, while the `temp_change_factor` was raised to promote faster temperature adaptations during failed transition tests. It is important to emphasize that while these changes may increase the search for solutions, they could lead to poorer paths being found. On the other hand, to ensure safer handling and reduce tilting during manipulation, the orientation constraint of the end-effector was restricted for the 400x300 boxes. This constraint was also adjusted for the placement code of the 600x400 boxes, as they are placed full on the upper level of the ramp. MoveIt includes post-processing algorithms to time parameterization kinematic trajectories, considering the velocity and acceleration values. To prevent jerky movements of the manipulator and improve smoothness, the Ruckig jerk-limited smoothing algorithm was used during the pick-and-place operations (PickNik Robotics, 2024d). The performance comparison between this custom T-RRT and the default one for the benchmarking metric across the different operations is depicted in Appendix G.

Another post-processing optimization technique that can be employed is using CHOMP as a post-processor for OMPL (PickNik Robotics, 2024e). In this approach, the custom T-RRT finds a solution, and CHOMP subsequently optimizes the trajectory. The CHOMP trajectory

initialization method `fillTrajectory` was selected to ensure that the computed path from T-RRT was received. Similarly to the previously considered process, the CHOMP parameters were modified iteratively to guarantee a solution in the shortest possible time. The maximum number of iterations after establishing a collision-free trajectory was reduced, the weight of smoothness in the cost function was considered, a minimum obstacle clearance was defined, and the `enable_failure_recovery` parameter was set to true, which enables CHOMP to adjust its parameters in case the first attempt with the specified parameters fails. The performance of this parameter is also illustrated in Appendix G.

For operations involving the 400x300 boxes, it was observed that the custom T-RRT could generate faster paths, even though the manipulator was more restricted. The execution time values were similar to the default planner, but the path length was shorter with more concentrated values, as indicated by the smaller interquartile range. This reduced variation in values may be attributed to restrictions imposed in the codes that force the algorithm to execute trajectories in a certain direction consistently. In both operations, the use of restricted orientation constraints contributed to smoother paths, particularly in the placement scenario, where the reduction was significant. Regarding the T-RRT with CHOMP, it was verified that although the OMPL quickly found initial solutions and the CHOMP parameters were adjusted, the post-processing time remained lengthy. The planning time ranged from 6 to 7 seconds, which is too long for practical real-world applications. For the smoothness metric, this planner achieved low values and, in the pick operation, even outperformed the custom T-RRT. Although not included in Appendix G, in the ten simulations conducted for each planner, both were able to find solutions, usually in one or two attempts. In one simulation of the pick operation, the planner T-RRT with CHOMP required six attempts to find a solution for the retreat movement to the home position. This result was reflected in the outlier in the planning time for the retreat movement in the respective operation.

Regarding the pick and place operations involving the larger boxes, the custom T-RRT planner generally demonstrated shorter planning times than the default planner. However, its execution times were slightly higher in both operations. In turn, the default planner exhibited higher mean execution times for the place operation due to the presence of outliers. The custom planner produced shorter trajectories for the path length metric in both operations. Concerning the smoothness metric, both planners displayed similar values for the pick operation, as the retreat phase consists of a short movement without any collision objects

in its path. Nonetheless, custom T-RRT had better results in the place operation. As previously discussed, this performance behavior may result in the restricted orientation constraint applied to this operation. The custom T-RRT with CHOMP planner accomplished results similar to those of the custom T-RRT. Nevertheless, it required longer planning times due to the post-processing step, resulting in values of around 6 to 7 seconds.

From all this analysis, it can be concluded that both the customized T-RRT planner and the planner in conjunction with CHOMP delivered better results in the smoothness metric for small box operations. This outcome was desired, as the default T-RRT planner exhibited high values for this metric. The planners had similar results for operations with larger boxes, although the two customized planners generated shorter paths than the default one. It is also evident that adding CHOMP as a post-processor did not significantly enhance the performance of the custom T-RRT. The results remained similar, with the additional drawback of increased planning time. This is likely attributed to the parameter adjustments made to CHOMP, where the concern was minimizing the post-processing time rather than improving the planner's smoothness. Therefore, based on the realized study, the customized T-RRT is the most suitable planner for the working environment present in the MIAR project.

5.4 Manipulation Operations Using the Selected Motion Planner

The customized T-RRT motion planner demonstrated the best performance, according to the studies conducted earlier. The comparison focused solely on scenarios where the boxes were stored on the access ramps due to the challenging nature of this environment for the manipulator during the tasks. However, this evaluation did not include the four additional scenarios where the robot picks and deposits the *Targets* in the dollies located at the Dolly Supply area. Therefore, the chosen motion planner was tested for the eight manipulation scenarios to ensure its suitability for the MIAR project. This analysis considered all possible pick-and-place operations involving the transport of the *Targets*. Unlike the previous benchmarking studies, these tests incorporated real-time communication between MoveIt and CoppeliaSim.

This analysis verified that the planning times occasionally exceeded expectations compared to the results obtained solely in MoveIt. This slight discrepancy can be attributed to the continuous communication between the two software systems, which imposes greater

computational demands on the system. Despite this, the planner successfully identified feasible solutions for the various case scenarios.

The chosen planner was determined to effectively accomplish all intended tasks with satisfactory results. Figure 84 shows the outcomes of the picking operations for the 600x400 boxes and 400x300 boxes from the ramp.

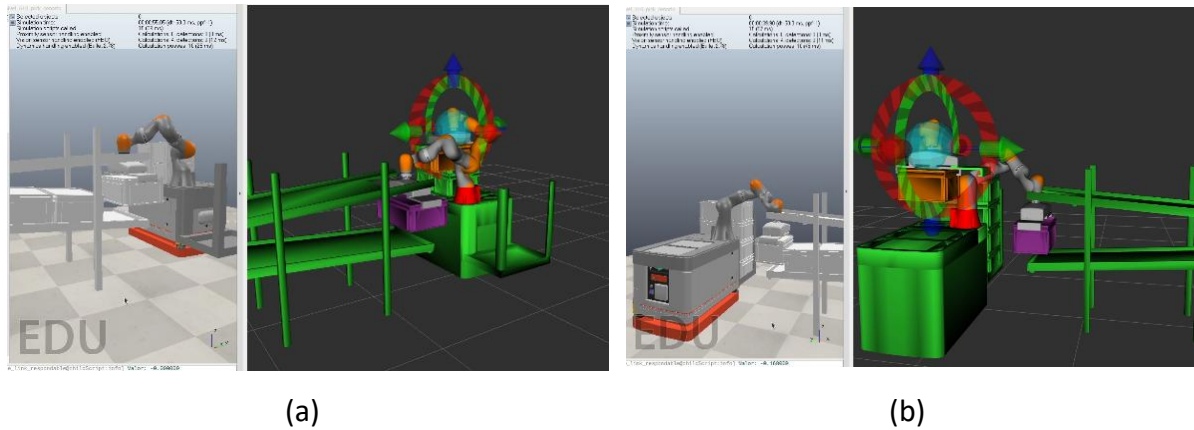


Figure 84 - (a) Picking 600x400 boxes from the ramp. The task can be seen in: https://youtu.be/4_dr845m8Tw; (b) Picking 400x300 boxes from the ramp. The task can be seen in: <https://youtu.be/8CKV2Hu8QAo>.

Similarly, the placement of the 600x400 boxes on the ramp, as well as the placement of the 400x300 boxes, can be checked in Figure 85.

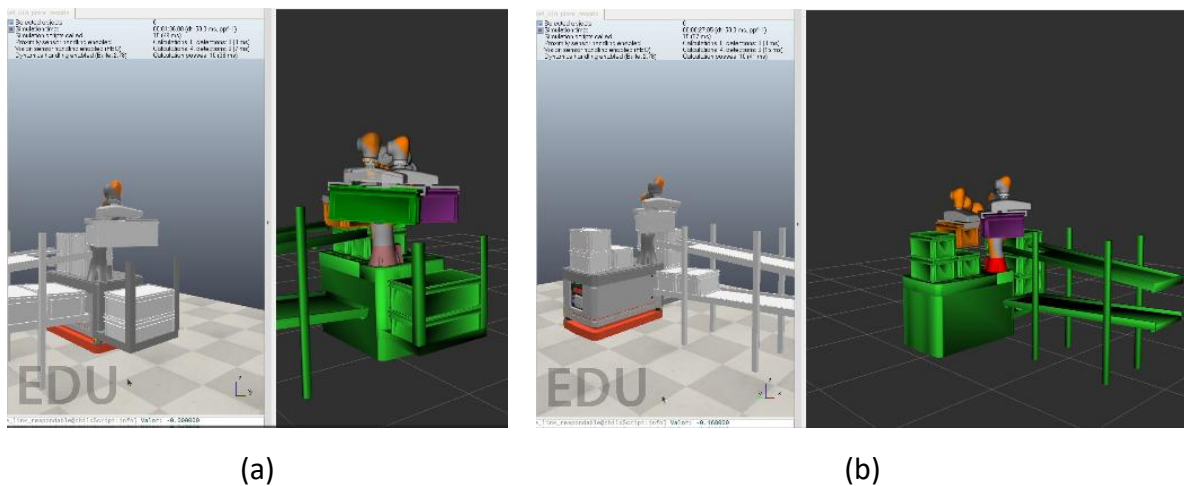


Figure 85 - (a) Placing 600x400 boxes in the ramp. The task can be seen in: https://youtu.be/a5ITRRm8M_8; (b) Placing 400x300 boxes in the ramp. The task can be seen in: <https://youtu.be/GSF92bFSylc>.

Finally, the operations conducted in the Dolly Supply area are depicted in Figure 86, showcasing both types of operations for the two box sizes.

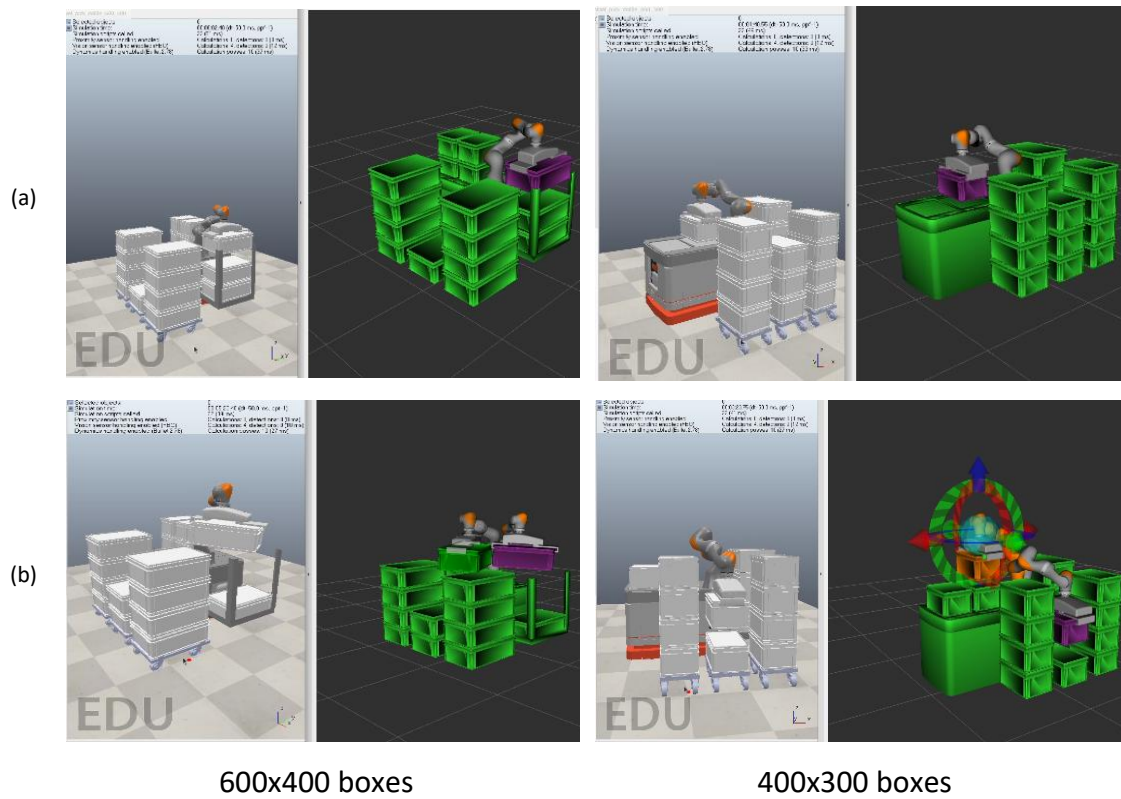


Figure 86 - (a) Picking the *Targets* from the Dolly Supply. The picking task for the large boxes can be seen in: <https://youtu.be/0hr2zuzmehs>. The picking task for the small boxes can be seen in: <https://youtu.be/aKFyQ0qsOxU>; (b) Placing the *Targets* in the Dolly Supply. The placing task for the large boxes can be seen in: <https://youtu.be/kp6llsDKCKc>. The placing task for the small boxes can be seen in: <https://youtu.be/3-6wfUqV1q8>.

6 Conclusions

As outlined at the beginning of this dissertation, the primary objective was to develop collision-free pick-and-place trajectories for the redundant manipulator KUKA LBR iiwa 14 R820, integrated into a mobile robot. The manipulator's tasks involved picking up full *Targets* from the Dolly Supply, placing them on access ramps, removing the empty boxes from the lower level of the ramps, and returning them to the storage buffer. As there are two different types of *Targets*, the 600x400 boxes and 400x300 boxes, it was necessary to perform eight different manipulation operations to achieve the goals of this work.

A digital twin of the manipulator's workspace at the Bosch Car Multimedia shop floor was successfully created in CoppeliaSim. This simulation scenario incorporated models of the *Targets*, dollies, dolly channels, and access ramps, which were initially designed in Autodesk Fusion and later imported. The scenario was used to validate the autonomous mobile manipulator's ability to execute the required operations. It was concluded that the arm could handle up to four *Targets* from the dollies, while the KMR could transport either four smaller boxes or two larger *Targets* at its base. To address the low storage capacity of the mobile platform, a conceptual structure capable of carrying eight *Targets*, four of each type, was designed and served as a practical solution for the manipulation tests, meeting the project's requirements.

Regarding the selection of the gripper, a detailed analysis using the weighted objectives method identified the collaborative 2FGP20 gripper as the most appropriate choice. Despite meeting the required requirements, the chosen gripper's height limited the manipulator's ability to handle the maximum *Target* weight of 8 kg. Custom gripper fingers were designed and simulated using a linear static analysis to endure such loads and withstand the maximum gripping force. The analysis revealed that the finger more susceptible to high-stress values was the movable finger, primarily due to its greater length. Although the fingers appeared to be over-dimensioned, their shape optimization was postponed, given their provisional nature in the project's initial phase.

It was determined that the current access ramps make it impossible to place or collect the *Targets*. A new design featuring a cut in the front area of the upper level was proposed.

This study was critical to ensure the manipulator retreats from the ramps after picking up the boxes on the lower level.

A key objective to consider in the project is that the movement of the manipulator must be legible and smooth during operations, given that KMR will be working in an area surrounded by human operators. Although these operators will not directly interact with the manipulator, they must be aware of its actions. Thereby, joint and orientation constraints were implemented in MoveIt to ensure proper handling and control of the arm. Given the variety of motion algorithms available in MoveIt, a benchmarking study was conducted on the scenarios that present more significant challenges for the arm, which are the tasks of picking and placing the boxes on the ramps. The comparison revealed that only three planners could guarantee solutions across the four manipulation scenarios, which were KPIECE, LazyPRM*, and T-RRT, with the latter achieving the best results. This planner was further customized to enhance trajectory smoothness while maintaining short planning times. CHOMP was also used as a post-processor to improve the results. However, it was concluded that the outcomes were similar, with T-RRT emerging as the most effective planner.

An innovative communication method between CoppeliaSim and MoveIt was developed, leveraging ROS2 functionalities to connect these software. While MoveIt was used with the purpose of generating trajectories for the robotic arm because it enshrines a set of planning pipelines such as CHOMP, PILZ, and OMPL, CoppeliaSim was employed to simulate all physical interactions during the simulation. This accomplishment benefits robotics by providing a viable way for communicating between the two software. This approach could be useful in future research instead of resorting to the traditional MoveIt-Gazebo connection. Furthermore, it enables the application of new planning algorithms to CoppeliaSim for trajectory generation, rather than limiting CoppeliaSim to only using the OMPL library via a plugin.

Overall, the developed project successfully achieved its main objective of developing safe and smooth movements for a manipulator during logistics tasks. The intensive analysis performed to select the most suitable planner for the project can serve as a future guideline in research projects with similar manipulator workspaces.

6.1 Future work

Albeit the goals reached in this dissertation represent a significant milestone for the initial phase of the MIAR project, many factors still require further analysis, and certain limitations need to be overcome. Next, several suggestions are outlined for consideration in potential future studies.

It is important to recognize that not all goals were fully met. In particular, selecting the gripper for the arm presented challenges, as no solution currently fulfills all the system's requirements. To cope with this problem, the development of a custom gripper should not be discarded. This approach would present a challenge for the project, as the gripper would need to be lightweight, capable of handling the maximum weight required for the boxes, and have small dimensions to avoid affecting the arm's payload. Furthermore, the development of optimized fingers should also be considered in the advanced stages of the project.

Alternatively, exploring a different arm for the project could open a broader range of available grippers on the market, reducing the need for custom development. The arm would need to be redundant to handle the challenging pick-and-place tasks while maintaining the quality of generated paths and avoiding obstacles. It would also need to be compatible with an omnidirectional platform with dimensions less than 1200 mm to navigate the narrow aisles of the shop floor with sufficient maneuvering space.

The integration with the Task Manager, the submodule responsible for issuing commands for the various robotic tasks of the project, constitutes a crucial subsequent step. The Arm Planning submodule must be capable of receiving the planning order and the object to manipulate from the Task Manager. To enable the arm to grasp a specific object, the object must first be detected and identified, and collision objects like the dollies and the ramps must be recognized. Therefore, a robust connection between the Arm Planning submodule and the Vision System submodule is essential. These interconnections will significantly contribute to realizing the previously outlined global architecture.

The development of pick-and-place codes, as well as a similar study to the one presented in Chapter 5 should also be applied to the scenarios involving the two-level ramps, as it would open doors to integrating these types of ramps into the future of the project if such ramps become eventually necessary.

In terms of trajectory planning, other pipelines should be explored, such as the STOMP planner, which was not included in the current work, but it has already proven to produce better results than CHOMP, both in terms of planning time and smoothness, as can be inspected in (S. Liu & Liu, 2022). Like CHOMP, STOMP can also be integrated as a post-processor with the OMPL planners in MoveIt.

Although sampling-based planners such as those from OMPL can find solutions quickly in complex environments, they tend not to be deterministic, meaning that different trajectories can be yielded for the same initial conditions and start and goal configurations. Regardless of how well-suited the chosen planner attends to be, this unpredictability presents a challenge in logistics projects, such as the case study presented at Bosch, where reliability is crucial. To address this, pre-planned paths tailored for each specific task could be employed. This change would eliminate planning times that exceed expectations and less successful trajectories. As a result, the planner's selection would not focus on finding a quick solution but instead prioritize generating high-quality paths. Asymptotically optimal planners like RRT*, PRM*, and LazyPRM* would be more relevant, as their convergence towards an optimal solution can be slow. Optimization-based planners such as CHOMP could be enhanced by adjusting the weights of the cost functions to prioritize smoothness and obstacle avoidance. Precision issues may arise in practical cases since the arm is positioned on top of a platform, making the pre-planned path insufficient for safe manipulation. Employing a hybrid approach would be beneficial for the project to avoid potential collisions in real-time. Incorporating a local planner is crucial during the phases of gripping the boxes and positioning the arm onto the lower level of the ramp.

References

- Alandes, C. (2015). *A comparison among different sampling-based planning techniques*. POLITECNICO DI MILANO.
- Alberto, C., & Borges, P. (2022). *Universidade do Minho Escola de Engenharia Inspection of Deformable Objects via Robotic Manipulation: a Learning from Demonstration Approach*. <https://hdl.handle.net/1822/81373>
- Albonico, M., Đorđević, M., Hamer, E., & Malavolta, I. (2023). Software engineering research on the Robot Operating System: A systematic mapping study. *Journal of Systems and Software*, 197, 111574. <https://doi.org/10.1016/j.jss.2022.111574>
- Amadi, C., Mbanisi, K., & Smit, W. (2024). *An Introduction to the ros2_control framework using a low cost Differential Drive Robot*. <https://doi.org/10.13140/RG.2.2.15748.54408>
- Annem, V., Rajendran, P., Thakar, S., & Gupta, S. K. (2019, September). Towards Remote Teleoperation of a Semi-Autonomous Mobile Manipulator System in Machine Tending Tasks. *Volume 1: Additive Manufacturing; Manufacturing Equipment and Systems; Bio and Sustainable Manufacturing*. <https://doi.org/10.1115/MSEC2019-3027>
- Aslan. (2021, March 2). *Local Path Planning Using virtual Potential Field in Python*.
- Azadeh, K., De Koster, R., & Roy, D. (2019). Robotized and Automated Warehouse Systems: Review and Recent Developments. *Transportation Science*, 53(4), 917–945. <https://doi.org/10.1287/trsc.2018.0873>
- B. Sciavicco, L. V. L., & Oriolo, G. S. (2009). *Robotics: Modelling, planning and control* (1 edition). Springer.
- Barland, M. (2012). *Motion Planning Framework for Industrial Manipulators using the Open Motion Planning Library (OMPL)*. <https://api.semanticscholar.org/CorpusID:55053992>
- Bogaerts, B., Sels, S., Vanlanduit, S., & Penne, R. (2020). Connecting the CoppeliaSim robotics simulator to virtual reality. *SoftwareX*, 11, 100426. <https://doi.org/10.1016/j.softx.2020.100426>
- Bohlin, R., & Kavraki, L. (2000). Path planning using lazy PRM. *IEEE Int Conf Robotics and Automation*, 1, 521–528 vol.1. <https://doi.org/10.1109/ROBOT.2000.844107>

- Chitta, S., Sucan, I., & Cousins, S. (2012). MoveIt! [ROS Topics]. *IEEE Robotics & Automation Magazine*, 19(1), 18–19. <https://doi.org/10.1109/MRA.2011.2181749>
- Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., & Thrun, S. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press.
- Cohen, B., Sucan, I. A., & Chitta, S. (2012). A generic infrastructure for benchmarking motion planners. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 589–595. <https://doi.org/10.1109/IROS.2012.6386228>
- Coleman, D., Sucan, I., Chitta, S., & Correll, N. (2014). *Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study*.
- Considine, D. M., & Considine, G. D. (1986). *Robot Technology Fundamentals*. <https://api.semanticscholar.org/CorpusID:59927231>
- Craig, J. (2017). *Introduction to Robotics: Mechanics and Control* (4th ed.). Pearson.
- da Costa, P. G. (2019). *Motion Planning in Cartesian Space for the Collaborative Redundant Robot Sawyer*.
- Devaurs, D., Simeon, T., & Cortes, J. (2013). Enhancing the transition-based RRT to deal with complex cost spaces. *2013 IEEE International Conference on Robotics and Automation*, 4120–4125. <https://doi.org/10.1109/ICRA.2013.6631158>
- Dheenadhayalan. (2021). *fusion2urdf ROS2*. GitHub. <https://github.com/dheena2k2/fusion2urdf-ros2?tab=readme-ov-file>
- Doyle-Kent, M., & Kopacek, P. (2021). Adoption of Collaborative Robotics in Industry 5.0. An Irish industry case study. *IFAC-PapersOnLine*, 54(13), 413–418. <https://doi.org/10.1016/j.ifacol.2021.10.483>
- Dragan, A. D. (2015). *Legible Robot Motion Planning*. <https://api.semanticscholar.org/CorpusID:62974192>
- Elbanhawi, M., & Simic, M. (2014). Sampling-Based Robot Motion Planning: A Review. *IEEE Access*, 2, 56–77. <https://doi.org/10.1109/ACCESS.2014.2302442>
- Engemann, H., Du, S., Kallweit, S., Cönen, P., & Dawar, H. (2020). OMNIVIL—An Autonomous Mobile Manipulator for Flexible Production. *Sensors*, 20(24), 7249. <https://doi.org/10.3390/s20247249>

- Eric O. Scott. (2015, October 25). *PRM with Ob-maps.gif*.
https://en.wikipedia.org/wiki/Probabilistic_roadmap#/media/File:PRM_with_Ob-maps.gif
- Faria, C., Ferreira, F., Erhagen, W., Monteiro, S., & Bicho, E. (2018). Position-based kinematics for 7-DoF serial manipulators with global configuration control, joint limit and singularity avoidance. *Mechanism and Machine Theory*, 121.
<https://doi.org/10.1016/j.mechmachtheory.2017.10.025>
- Fox, D., Burgard, W., & Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1), 23–33.
<https://doi.org/10.1109/100.580977>
- Gasparetto, A., & Zanotto, V. (2010). Optimal trajectory planning for industrial robots. *Advances in Engineering Software*, 41(4), 548–556.
<https://doi.org/10.1016/j.advengsoft.2009.11.001>
- Gomes De Brito, J. N. (2018). *Manipulador robótico para poda automática (Projeto RoMoVi)*.
- Groover, M. P. (2007). *Automation, Production Systems, and Computer-Integrated Manufacturing* (3rd ed.). Prentice Hall Press.
- Guertler, M., Tomidei, L., Sick, N., Carmichael, M., Paul, G., Wambsganss, A., Hernandez Moreno, V., & Hussain, S. (2023). WHEN IS A ROBOT A COBOT? MOVING BEYOND MANUFACTURING AND ARM-BASED COBOT MANIPULATORS. *Proceedings of the Design Society*, 3, 3889–3898. <https://doi.org/10.1017/pds.2023.390>
- Gulletta, G., Erhagen, W., & Bicho, E. (2020). Human-Like Arm Motion Generation: A Review. *Robotics*, 9(4), 102. <https://doi.org/10.3390/robotics9040102>
- Gulletta, G., Silva, E. C. E., Erhagen, W., Meulenbroek, R., Costa, M., & Bicho, E. (2021). A Human-like Upper-limb Motion Planner: Generating naturalistic movements for humanoid robots. *International Journal of Advanced Robotic Systems*, 18, 172988142199858. <https://doi.org/10.1177/1729881421998585>
- Hai Robotics. (2022). *ACR SYSTEMS Power Different Sectors*.
- Hauser, K. (2015). Lazy collision checking in asymptotically-optimal motion planning. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2951–2957.
<https://doi.org/10.1109/ICRA.2015.7139603>

- Hollerbach, J. M. (1985). *OPTIMUM KINEMATIC DESIGN FOR A SEVEN DEGREE OF FREEDOM MANIPULATOR*. <https://api.semanticscholar.org/CorpusID:10553087>
- Hsu, D., Latombe, J.-C., & Motwani, R. (1997). Path planning in expansive configuration spaces. *Proceedings of International Conference on Robotics and Automation*, 2719–2726. <https://doi.org/10.1109/ROBOT.1997.619371>
- Hwang, Y. K., & Ahuja, N. (1992). Gross motion planning—a survey. *ACM Computing Surveys*, 24(3), 219–291. <https://doi.org/10.1145/136035.136037>
- IFR International Federation of Robotics. (2023). *World Robotics 2023*.
- International Organization for Standardization. (2011a). Robots and robotic devices — Safety requirements for industrial robots Part 1: Robots. In *ISO Standard No. 10218-1:2011*. <https://www.iso.org/standard/51330.html>
- International Organization for Standardization. (2011b). Robots and robotic devices — Safety requirements for industrial robots Part 2: Robot systems and integration. In *ISO Standard No. 10218-2:2011*.
- International Organization for Standardization. (2016). Robots and robotic devices — Collaborative robots. In *ISO Standard No. 15066:2016*.
- Iossifidis, I., & Schoner, G. (2004). Autonomous reaching and obstacle avoidance with the anthropomorphic arm of a robotic assistant using the attractor dynamics approach. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, 4295–4300 Vol.5. <https://doi.org/10.1109/ROBOT.2004.1302393>
- Jaillet, L., Cortés, J., & Siméon, T. (2010). Sampling-Based Path Planning on Configuration-Space Costmaps. *IEEE Transactions on Robotics*, 26(4), 635–646. <https://doi.org/10.1109/TRO.2010.2049527>
- Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., & Schaal, S. (2011). STOMP: Stochastic trajectory optimization for motion planning. *2011 IEEE International Conference on Robotics and Automation*, 4569–4574. <https://doi.org/10.1109/ICRA.2011.5980280>
- Kang, G., Kim, Y. B., Lee, Y. H., Oh, H. S., You, W. S., & Choi, H. R. (2019). Sampling-based motion planning of manipulator with goal-oriented sampling. *Intelligent Service Robotics*, 12(3), 265–273. <https://doi.org/10.1007/s11370-019-00281-y>

- Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7), 846–894. <https://doi.org/10.1177/0278364911406761>
- Kavraki, L. E., Svestka, P., Latombe, J.-C., & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4), 566–580. <https://doi.org/10.1109/70.508439>
- Kavraki Lab. (2024). *TRRT Class Reference*. OMPL. https://ompl.kavrakilab.org/classompl_1_1geometric_1_1TRRT.html#details
- Khatib, O. (1986). Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. In *Autonomous Robot Vehicles* (pp. 396–404). Springer New York. https://doi.org/10.1007/978-1-4613-8997-2_29
- Kobayashi, M., & Motoi, N. (2022). Local Path Planning: Dynamic Window Approach With Virtual Manipulators Considering Dynamic Obstacles. *IEEE Access*, 10, 17018–17029. <https://doi.org/10.1109/ACCESS.2022.3150036>
- Koenig, N., & Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, 2149–2154. <https://doi.org/10.1109/IROS.2004.1389727>
- KUKA. (2019). *LBR iiwa Operating Instructions*.
- KUKA AG. (2024). *KMR IIWA*. Kuka.Com. <https://www.kuka.com/en-de/products/amr-autonomous-mobile-robotics/mobile-robots/kmr-iiwa>
- Latombe, J. C. (1991). Robot Motion Planning. The Springer International Series in Engineering and Computer Science. In *Kluwer Academic* (1st ed). Kluwer Academic.
- Lauretti, C., Cordella, F., & Zollo, L. (2019). A Hybrid Joint/Cartesian DMP-Based Approach for Obstacle Avoidance of Anthropomorphic Assistive Robots. *International Journal of Social Robotics*, 11(5), 783–796. <https://doi.org/10.1007/s12369-019-00597-w>
- Lavalle, S., & Kuffner, J. (2000). Rapidly-Exploring Random Trees: Progress and Prospects. *Algorithmic and Computational Robotics: New Directions*.
- LaValle, S. M. (1998). Rapidly-exploring random trees : a new tool for path planning. *The Annual Research Report*. <https://api.semanticscholar.org/CorpusID:14744621>

- LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press.
<https://doi.org/10.1017/CBO9780511546877>
- Likhachev, M., & Ferguson, D. (2009). Planning Long Dynamically Feasible Maneuvers for Autonomous Vehicles. *The International Journal of Robotics Research*, 28(8), 933–945. <https://doi.org/10.1177/0278364909340445>
- Liu, S., & Liu, P. (2020, December). *A Review of Motion Planning Algorithms for Robotic Arm Systems*.
- Liu, S., & Liu, P. (2022). Benchmarking and optimization of robot motion planning with motion planning pipeline. *The International Journal of Advanced Manufacturing Technology*, 118. <https://doi.org/10.1007/s00170-021-07985-5>
- Liu, W., Chen, D., & Steil, J. (2017). Analytical Inverse Kinematics Solver for Anthropomorphic 7-DOF Redundant Manipulators with Human-Like Configuration Constraints. *Journal of Intelligent & Robotic Systems*, 86(1), 63–79. <https://doi.org/10.1007/s10846-016-0449-6>
- Lynch, K. M., & Park, F. C. (2017, March 3). *Modern Robotics - Mechanics, Planning, and Control*.
- Madier, D. (2021). *Practical Finite Element Analysis for Mechanical Engineers* (1st ed.). Dominique Madier .
- Maria, R., & Rodrigues, P. (2022). *Motion Planning for Autonomous Sorting Tasks using the Collaborative Sawyer Robot*. <https://hdl.handle.net/1822/87341>
- Matheson, E., Minto, R., Zampieri, E. G. G., Faccio, M., & Rosati, G. (2019). Human–Robot Collaboration in Manufacturing Applications: A Review. *Robotics*, 8(4), 100. <https://doi.org/10.3390/robotics8040100>
- Matúšová, M., Bučányová, M., & Hrušková, E. (2019). The future of industry with collaborative robots. *MATEC Web of Conferences*, 299, 02008. <https://doi.org/10.1051/matecconf/201929902008>
- McAtamney, L., & Corlett, E. N. (1993). RULA: a survey method for the investigation of work-related upper limb disorders. *Applied Ergonomics*, 24(2), 91–99. [https://doi.org/10.1016/0003-6870\(93\)90080-S](https://doi.org/10.1016/0003-6870(93)90080-S)
- mhubii. (2021). *lbr_fri_ros2_stack*. GitHub. https://github.com/lbr-stack/lbr_fri_ros2_stack

- mhubii. (2024). *lbr_ros2_control*. LBR-STACK. https://lbr-stack.readthedocs.io/en/latest/lbr_fri_ros2_stack/lbr_ros2_control/doc/lbr_ros2_control.html
- Moll, M., Sucan, I. A., & Kavraki, L. E. (2015). Benchmarking Motion Planning Algorithms: An Extensible Infrastructure for Analysis and Visualization. *IEEE Robotics & Automation Magazine*, 22(3), 96–102. <https://doi.org/10.1109/MRA.2015.2448276>
- NECULA, M., CÂMPEAN, E., & MORAR, L. (2022). DEFINING THE CHARACTERISTICS NEEDED FOR THE COBOTS DESIGN USED FOR THE SUPPLY OF CNC MACHINES . *Applied Mathematics, Mechanics, and Engineering*, 65(Special I), 171–177.
- Niermann, D., Petzoldt, C., & Freitag, M. (2023). Intuitive and Flexible Process Control for Autonomous Mobile Robots: A Case Study in a Large Logistics Enterprise. *Procedia CIRP*, 120, 153–158. <https://doi.org/10.1016/j.procir.2023.08.028>
- OnRobot. (2023). *DATASHEET 2FGP20*.
- Open Robotics. (2024a). *Understanding actions*. ROS2 Documentation: Humble. <https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Actions/Understanding-ROS2-Actions.html>
- Open Robotics. (2024b). *Understanding nodes*. ROS2 Documentation: Humble. <https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html>
- Open Robotics. (2024c). *Using event handlers*. ROS2 Documentation: Humble. <https://docs.ros.org/en/humble/Tutorials/Intermediate/Launch/Using-Event-Handlers.html#using-event-handlers>
- Palmieri, G., & Scoccia, C. (2021). Motion Planning and Control of Redundant Manipulators for Dynamical Obstacle Avoidance. *Machines*, 9(6), 121. <https://doi.org/10.3390/machines9060121>
- Peixoto, R. T. (2015). *Validação de design de armazém automatizado com uso de simulação discreta*.
- PHD Inc. (2001). *GRIPPER SELECTION GUIDE*.
- PickNik Robotics. (2024a). *Concepts*. MoveIt 2 Documentation. <https://moveit.picknik.ai/main/doc/concepts/concepts.html>
- PickNik Robotics. (2024b). *Move Group C++ Interface*. MoveIt 2 Documentation. <https://moveit.picknik.ai/humble/index.html>

- PickNik Robotics. (2024c). *Movelt Setup Assistant*. Movelt 2 Documentation. https://moveit.picknik.ai/humble/doc/examples/setup_assistant/setup_assistant_tutorial.html
- PickNik Robotics. (2024d). *Time Parameterization*. Movelt2 Documentation. https://moveit.picknik.ai/humble/doc/examples/time_parameterization/time_parameterization_tutorial.html
- PickNik Robotics. (2024e). *Using CHOMP Planner*. Movelt2 Documentation. https://moveit.picknik.ai/humble/doc/how_to_guides/chomp_planner/chomp_planner_tutorial.html
- PickNik Robotics. (2024f). *Using OMPL Constrained Planning*. Movelt 2 Documentation. https://moveit.picknik.ai/humble/doc/how_to_guides/using_ompl_constrained_planning/ompl_constrained_planning.html
- Quinlan, S., & Khatib, O. (1993). Elastic bands: connecting path planning and control. *[1993] Proceedings IEEE International Conference on Robotics and Automation*, 802–807. <https://doi.org/10.1109/ROBOT.1993.291936>
- Rafael, P., & Ferreira, R. (2022). *Solução robótica segura e flexível para alimentação de máquina ferramenta*.
- Rano, I., & Iossifidis, I. (2013). Modelling human arm motion through the attractor dynamics approach. *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2088–2093. <https://doi.org/10.1109/ROBIO.2013.6739777>
- Rastegarpanah, A., Gonzalez, H. C., & Stolkin, R. (2021). Semi-Autonomous Behaviour Tree-Based Framework for Sorting Electric Vehicle Batteries Components. *Robotics*, 10(2), 82. <https://doi.org/10.3390/robotics10020082>
- Robotics, C. (2023). *Robotize your manual warehouse*.
- Robotics, H. (2022). *ACR SYSTEMS Power Different Sectors*.
- Roodbergen, K. J., & Vis, I. F. A. (2009). A survey of literature on automated storage and retrieval systems. *European Journal of Operational Research*, 194(2), 343–362. <https://doi.org/10.1016/j.ejor.2008.01.038>
- ros2.org. (2020, October 9). *sensor_msgs/msg/JointState Message*. Ros2.Org. https://docs.ros2.org/foxy/api/sensor_msgs/msg/JointState.html
- Saoji, S., & Rosell, J. (2020). Flexibly configuring task and motion planning problems for mobile manipulators. *2020 25th IEEE International Conference on Emerging*

- Technologies and Factory Automation (ETFA)*, 1285–1288.
<https://doi.org/10.1109/ETFA46521.2020.9212086>
- Serrano, D. (2015). Introduction to ROS-Robot Operating System. *NATO Science & Technology Organization*.
- Shimizu, M., Kakuya, H., Yoon, W.-K., Kitagaki, K., & Kosuge, K. (2008). Analytical Inverse Kinematic Computation for 7-DOF Redundant Manipulators With Joint Limits and Its Application to Redundancy Resolution. *IEEE Transactions on Robotics*, 24(5).
<https://doi.org/10.1109/TRO.2008.2003266>
- Sucan, I. (2017). *moveit::planning_interface::MoveGroup Class Reference*. ROS.Org.
https://docs.ros.org/en/jade/api/moveit_ros_planning_interface/html/classmoveit_1_1planning__interface_1_1MoveGroup.html
- Sucan, I. A., Moll, M., & Kavraki, L. E. (2012). The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4), 72–82.
<https://doi.org/10.1109/MRA.2012.2205651>
- Sucan, I., & Kavraki, L. (2008). *Kinodynamic Motion Planning by Interior-Exterior Cell Exploration*. 449–464.
- Tavares, P. (2015). *Planeamento de trajetórias em Manipuladores em ambientes industriais*. Faculdade de Engenharia da Universidade do Porto.
- Thakar, S., Fang, L., Shah, B., & Gupta, S. (2018). Towards Time-Optimal Trajectory Planning for Pick-and-Transport Operation with a Mobile Manipulator. *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, 981–987. <https://doi.org/10.1109/COASE.2018.8560446>
- Thakar, S., Rajendran, P., Kabir, A. M., & Gupta, S. K. (2022). Manipulator Motion Planning for Part Pickup and Transport Operations From a Moving Base. *IEEE Transactions on Automation Science and Engineering*, 19(1), 191–206.
<https://doi.org/10.1109/TASE.2020.3020050>
- The Robotics Back-End. (2018). *What is a ROS Node?* <https://roboticsbackend.com/what-is-a-ros-node/>
- Toktaş-Palut, P. (2022). Analyzing the effects of Industry 4.0 technologies and coordination on the sustainability of supply chains. *Sustainable Production and Consumption*, 30, 341–358. <https://doi.org/10.1016/j.spc.2021.12.005>

- Trost, P., Kartnig, G., & Eder, M. (2022, September). *Simulation study of Autostore-systems*. <https://doi.org/10.34726/3044>
- Utz Group. (2014). *RAKO utz clever-creative-sustainable-relevant*.
- Vemula, A., Muelling, K., & Oh, J. (2021). Path Planning in Dynamic Environments with Adaptive Dimensionality. *Proceedings of the International Symposium on Combinatorial Search*, 7(1), 107–115. <https://doi.org/10.1609/socs.v7i1.18386>
- Weidemann, C., Mandischer, N., van Kerkom, F., Corves, B., Hüsing, M., Kraus, T., & Garus, C. (2023). Literature Review on Recent Trends and Perspectives of Collaborative Robotics in Work 4.0. *Robotics*, 12(3), 84. <https://doi.org/10.3390/robotics12030084>
- Wesley Reis, K., Antônio dos Reis, M., Tourinho Lima, R., & Alves Ferreira Neto, N. (2019). *ESTUDO DE DESEMPENHO DE PLANEJADORES DO MOVEIT! PARA UM MANIPULADOR ROBÓTICO DE 6DOF*.
- Xie, B., Zhao, J., & Liu, Y. (2011). Human-like motion planning for robotic arm system. *2011 15th International Conference on Advanced Robotics (ICAR)*, 88–93. <https://doi.org/10.1109/ICAR.2011.6088543>
- Xing, B., Gao, W.-J., Battle, K., Marwala, T., & Nelwamondo, F. (2010). *Simulation and optimization for batch order picking problem: the application of ant colony algorithm*. 1661–1672.
- Yang, S., Liu, P., & Pears, N. (2023). Benchmarking of Robot Arm Motion Planning in Cluttered Environments. *2023 28th International Conference on Automation and Computing (ICAC)*, 1–6. <https://doi.org/10.1109/ICAC57885.2023.10275283>
- Yang, Y., Merkt, W., Ivan, V., & Vijayakumar, S. (2018). Planning in Time-Configuration Space for Efficient Pick-and-Place in Non-Static Environments with Temporal Constraints. *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, 1–9. <https://doi.org/10.1109/HUMANOIDS.2018.8624989>
- Zacharias, F., Schlette, C., Schmidt, F., Borst, C., Rossmann, J., & Hirzinger, G. (2011). Making planned paths look more human-like in humanoid robot manipulation planning. *2011 IEEE International Conference on Robotics and Automation*, 1192–1198. <https://doi.org/10.1109/ICRA.2011.5979553>
- Zucker, M., Ratliff, N., Dragan, A. D., Pivtoraiko, M., Klingensmith, M., Dellin, C. M., Bagnell, J. A., & Srinivasa, S. S. (2013). CHOMP: Covariant Hamiltonian optimization

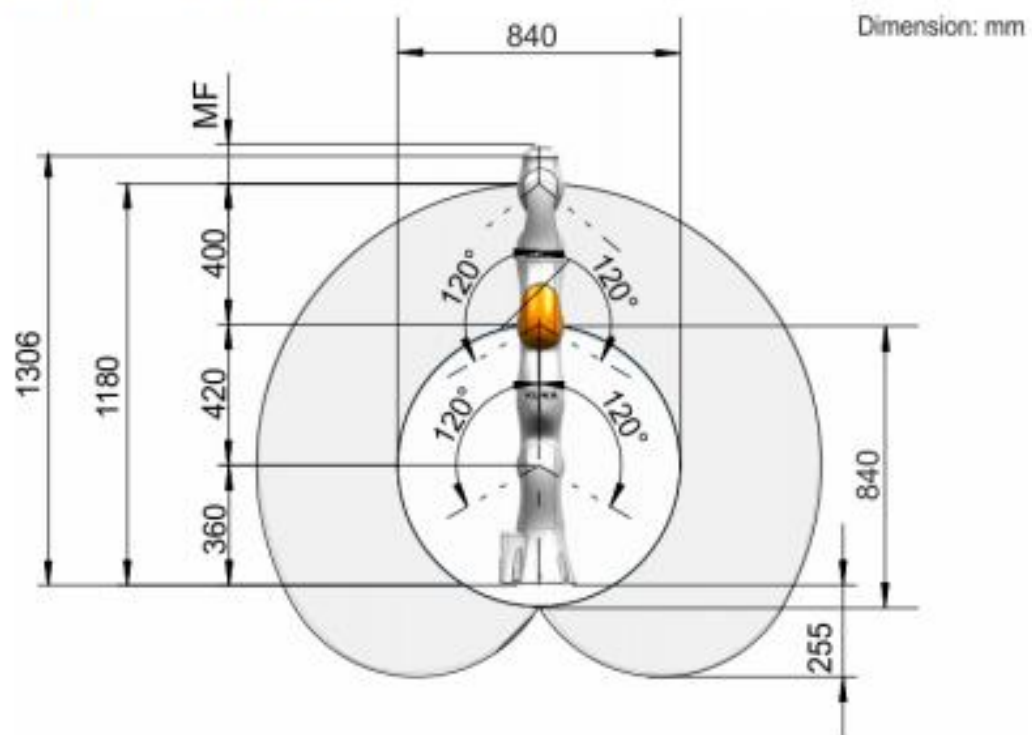
for motion planning. *The International Journal of Robotics Research*, 32(9–10), 1164–1193. <https://doi.org/10.1177/0278364913488805>

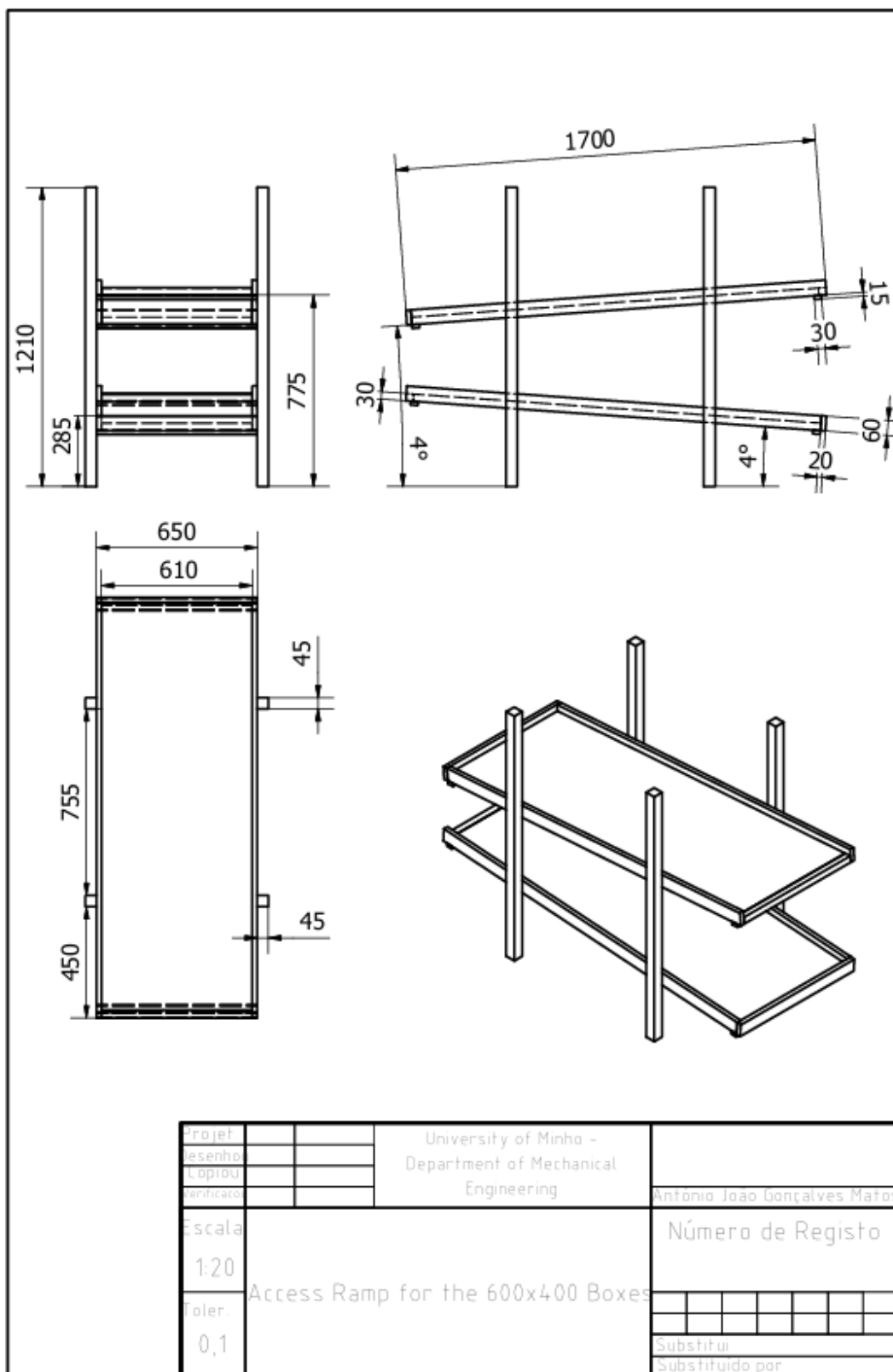
Annex

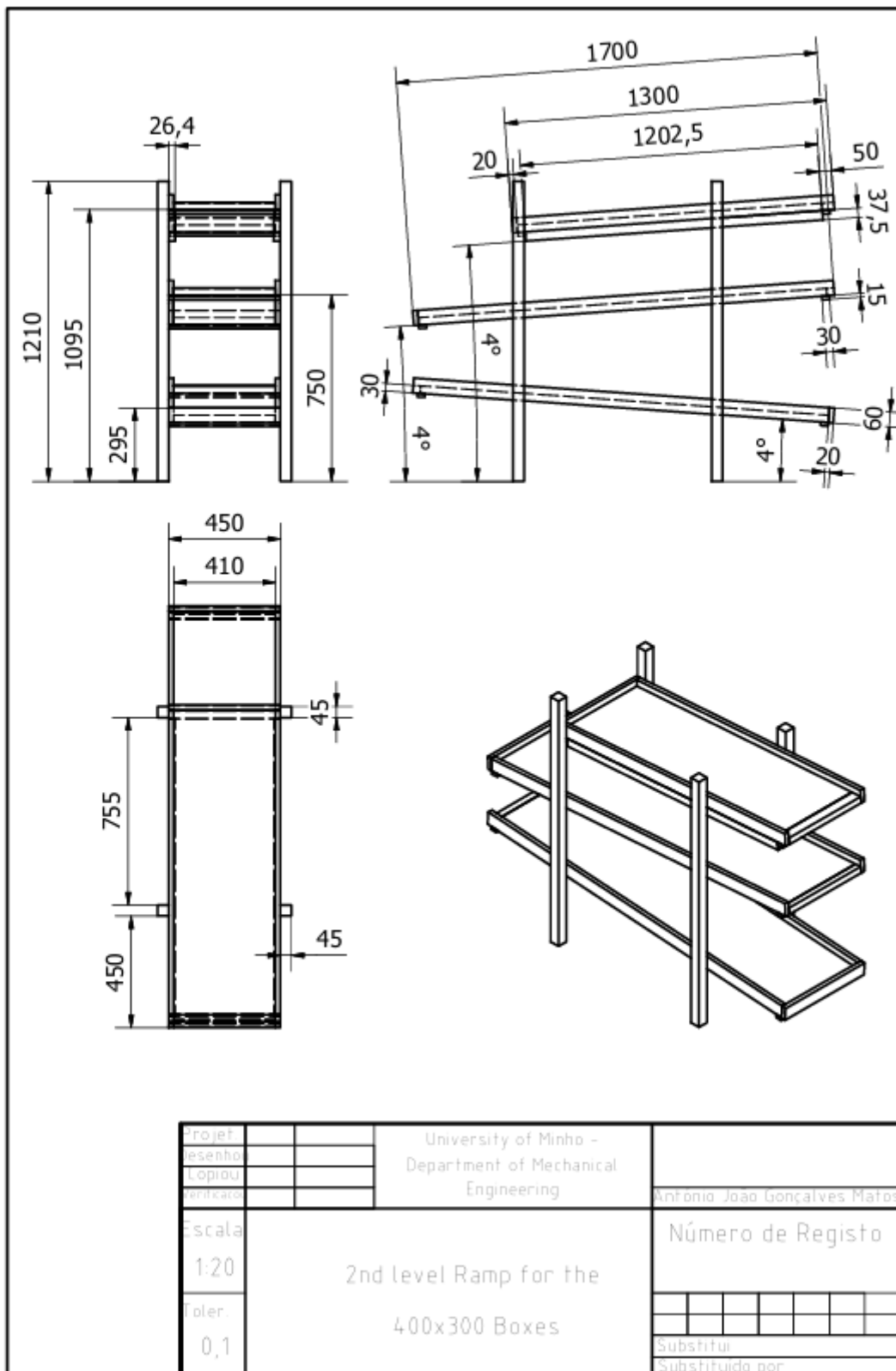
A. Workspace Graphic for the KUKA LBR iiwa 14 R820

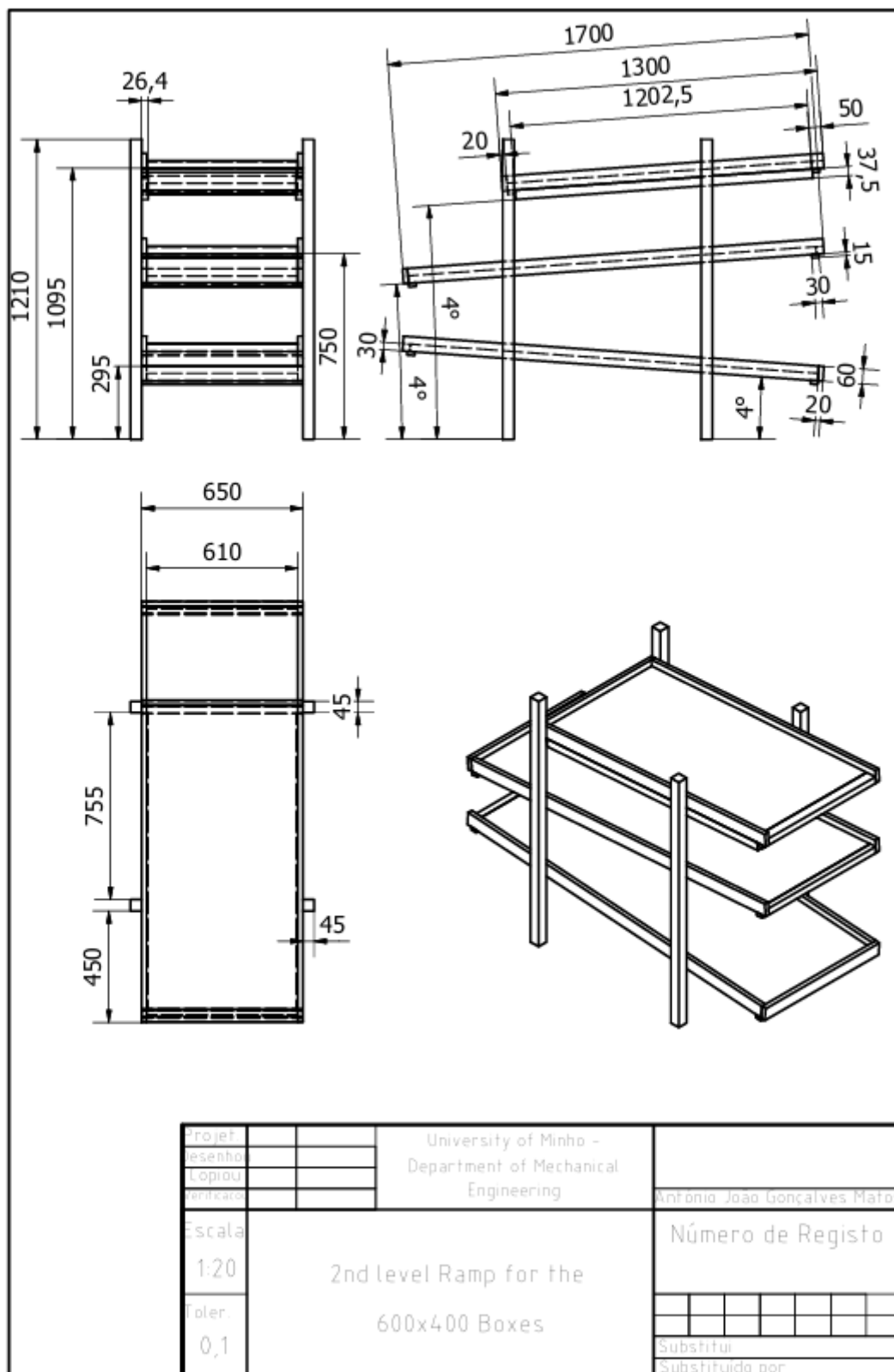
The present annex was adapted from (KUKA, 2019).

Workspace graphic

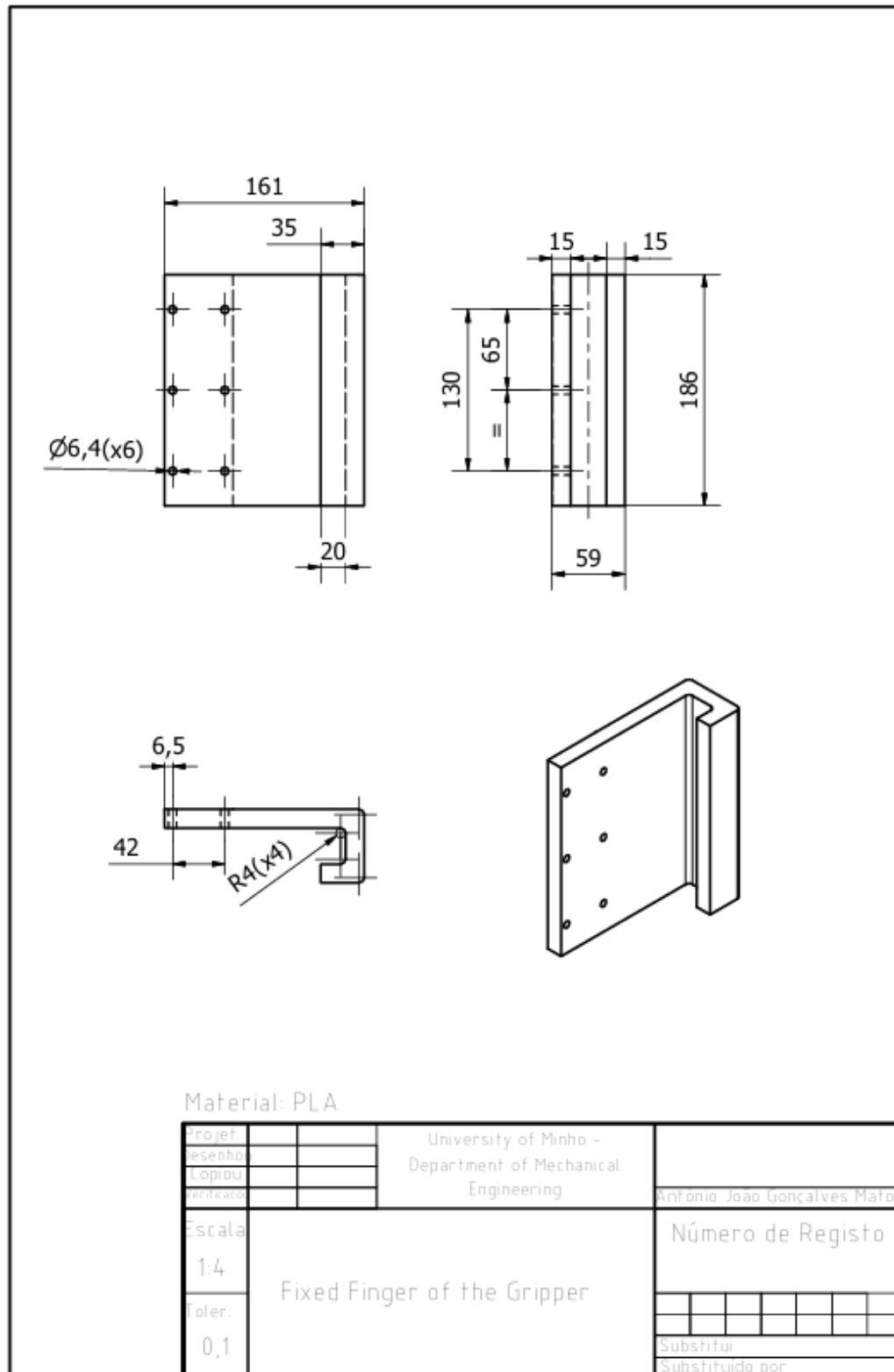








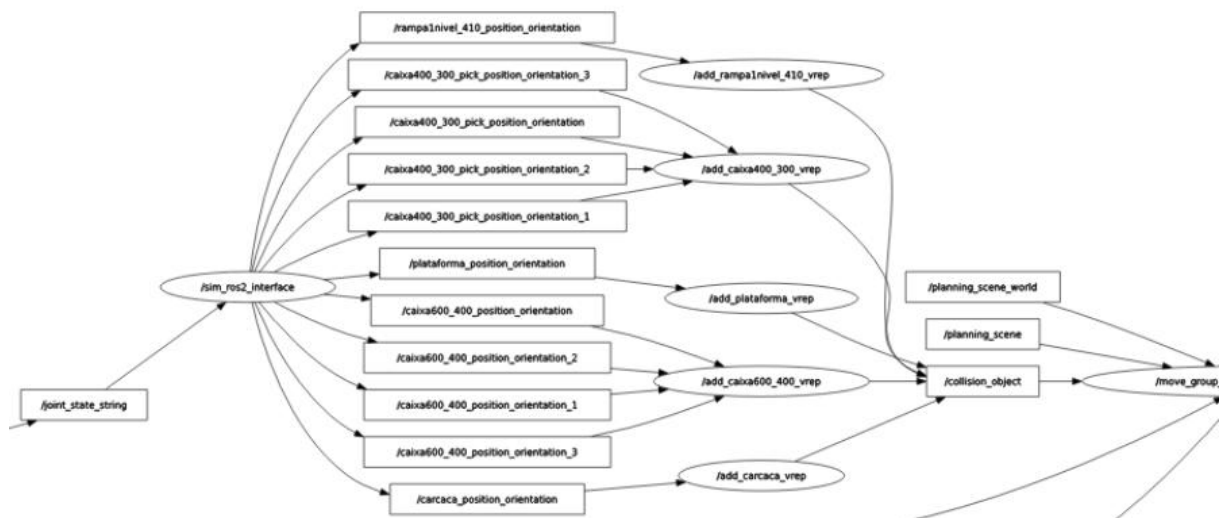
B. Technical Drawings of the Developed Fingers



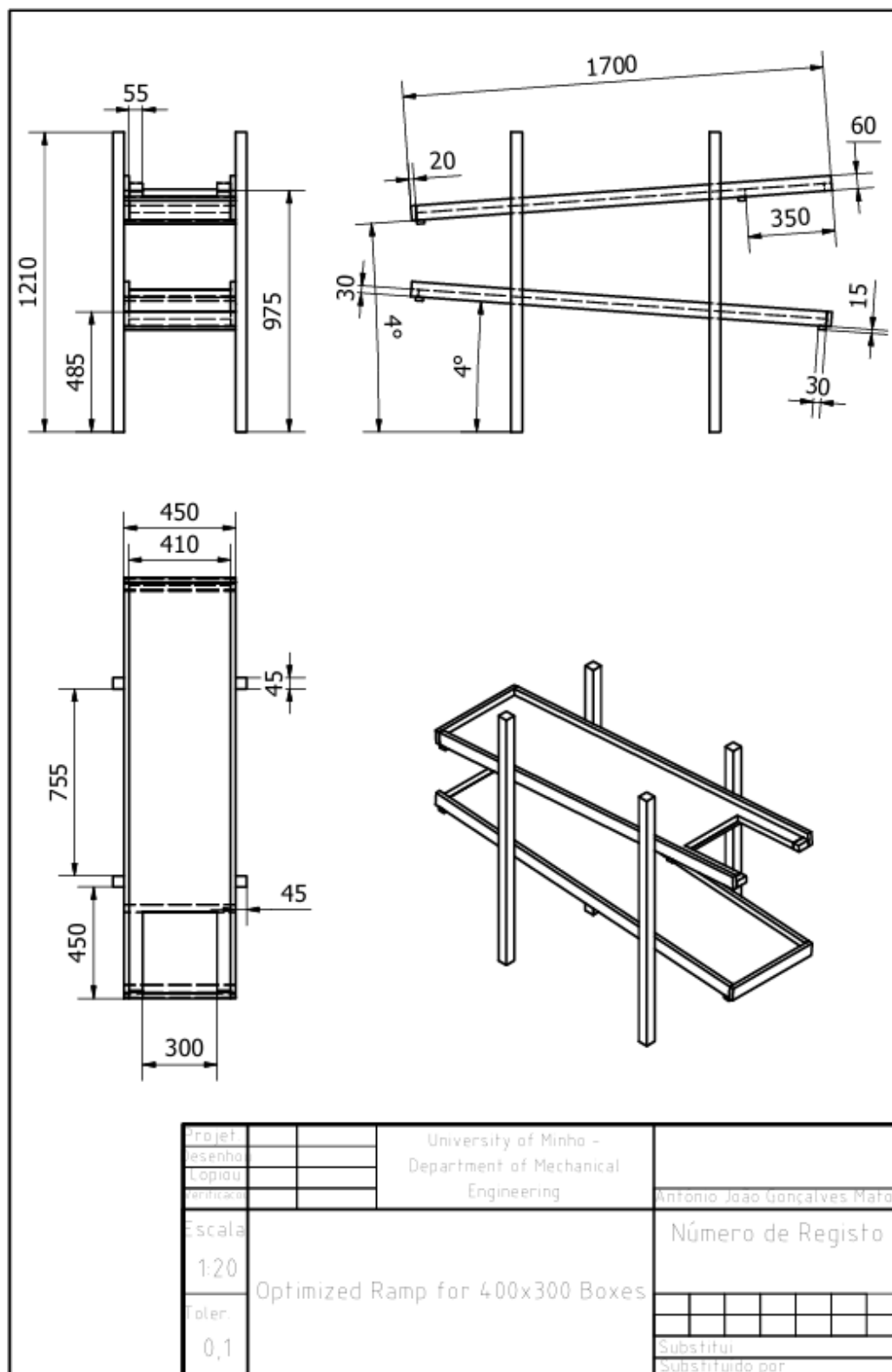
C. Nodes and Topics Responsible for the Transmission of the Manipulator's Joint States Between MoveIt and CoppeliaSim

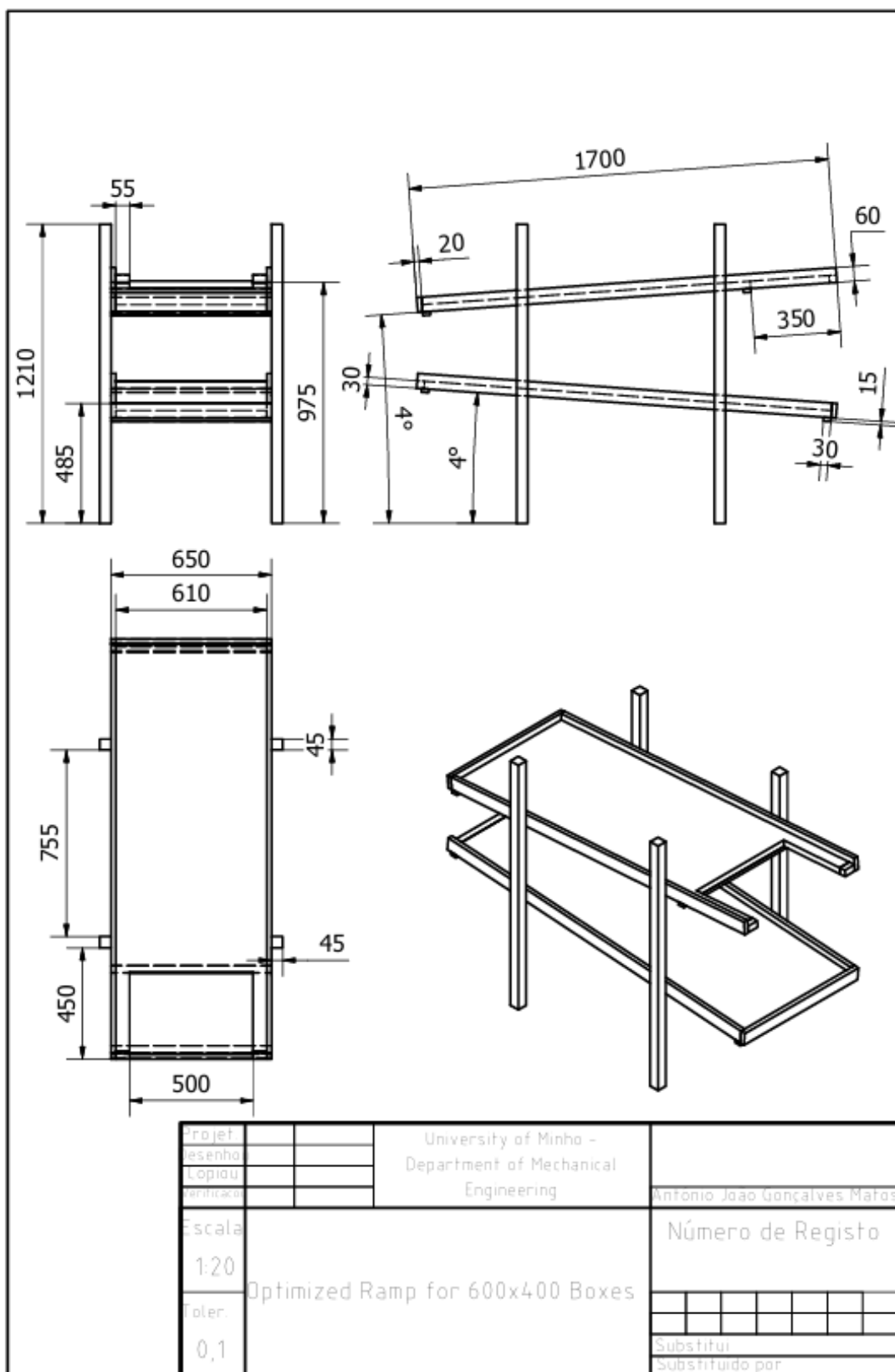


D. Nodes and Topics Responsible for the Transmission of the Objects' Poses Between MoveIt and CoppeliaSim



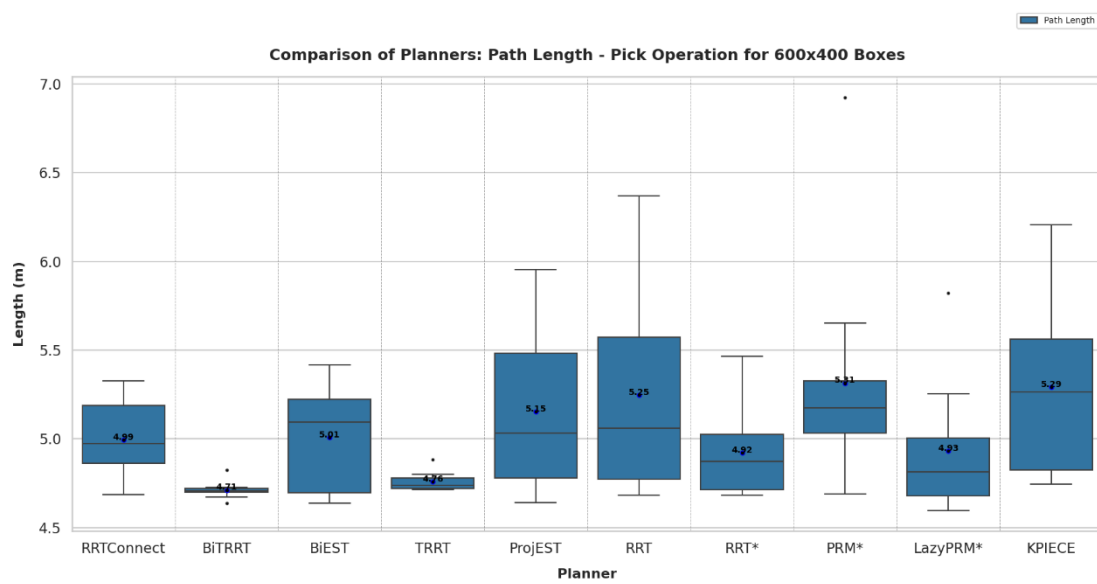
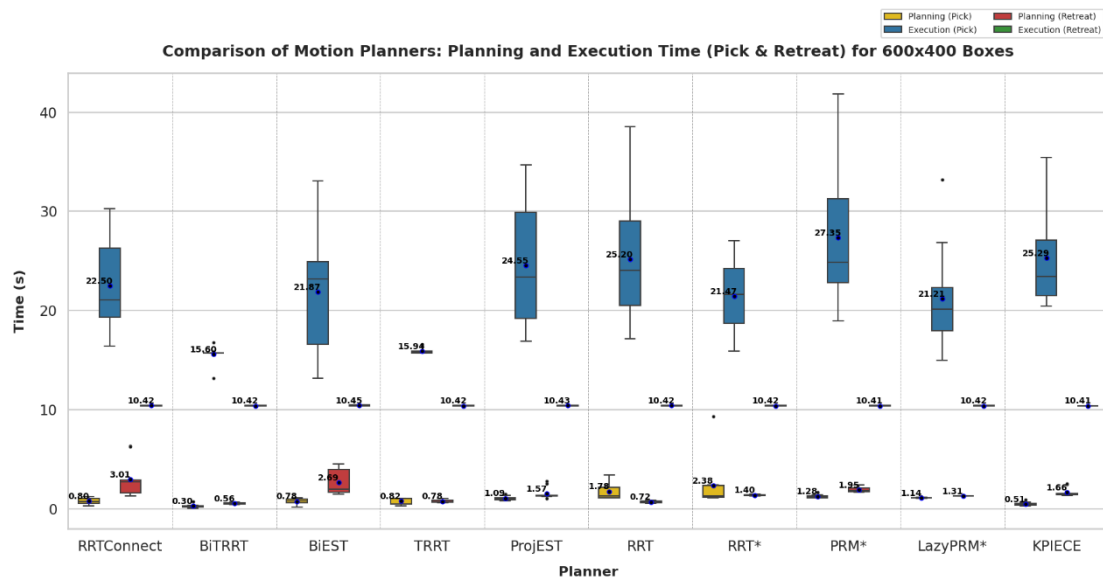
E. Technical Drawings of the “To Be” Access Ramps

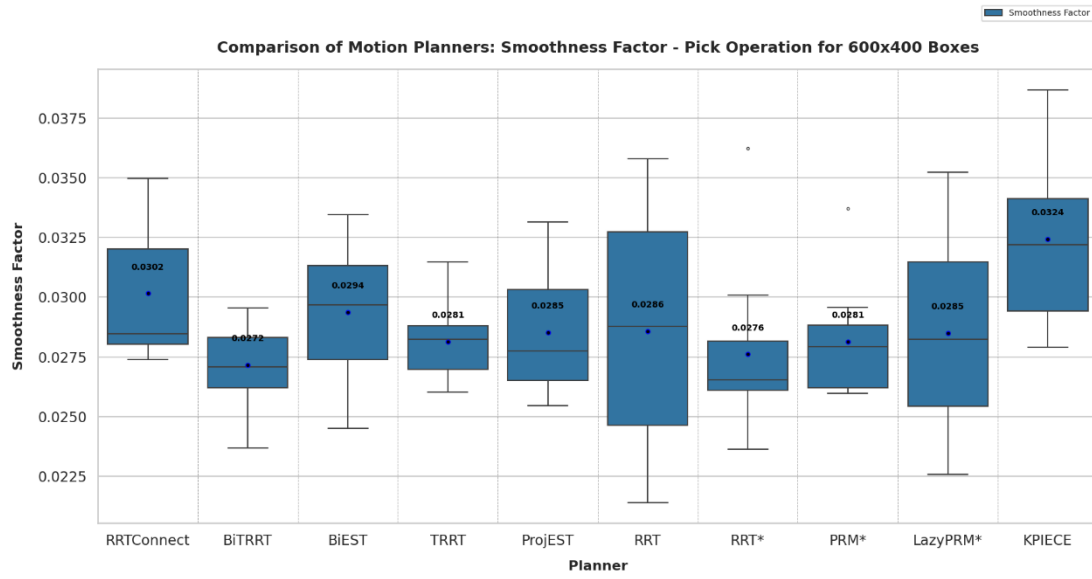




F. Benchmarking Study of the 10 OMPL planners

Pick Operation involving the 600x400 boxes

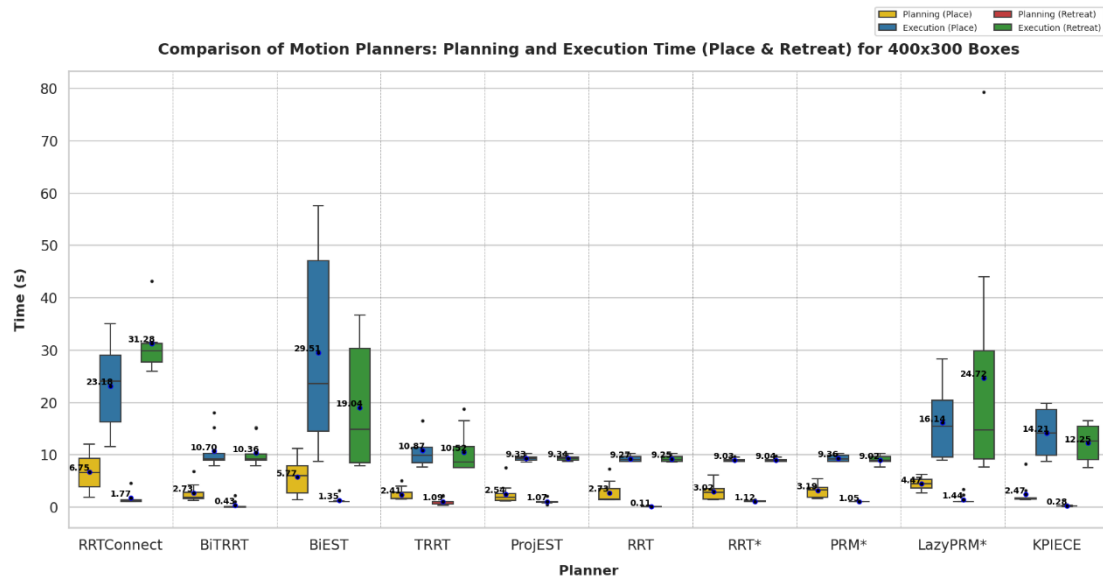


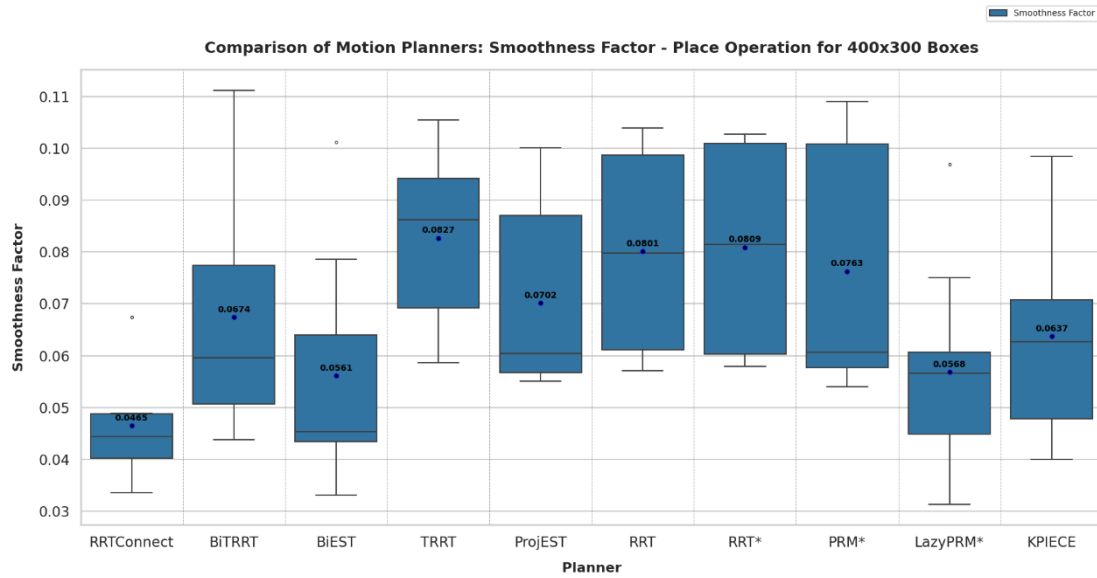
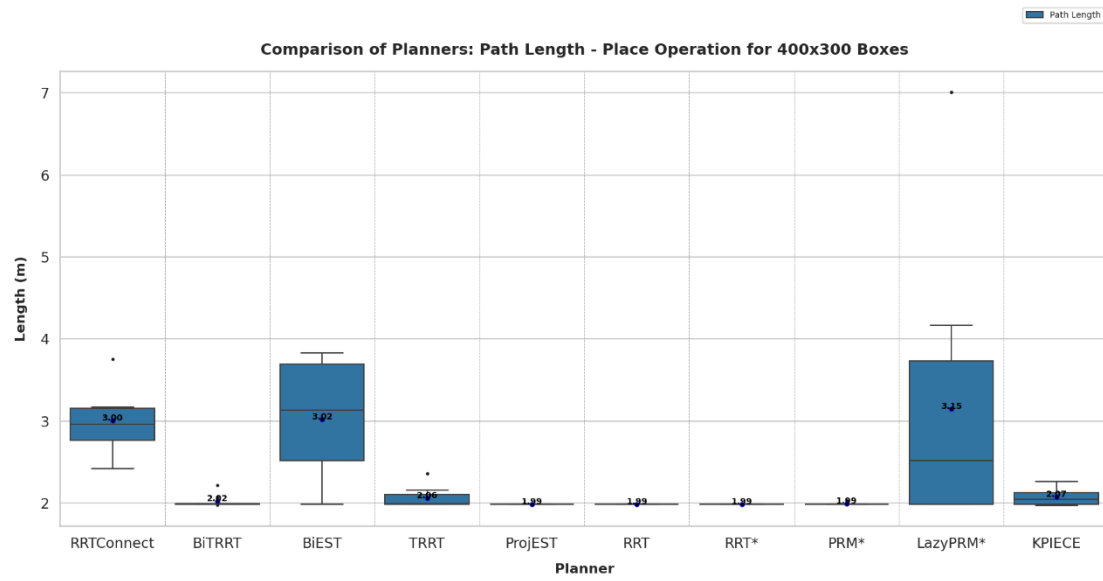


N° Attempts	RRT-Connect		Bi-TRRT		BIEST		T-TRT		ProjEST		RRT		RRT*		PRM*		LazyPRM*		KPIECE	
	pick	retreat	pick	retreat	pick	retreat	pick	retreat	pick	retreat	pick	retreat	pick	retreat	pick	retreat	pick	retreat	pick	retreat
1	10	4	10	10	10	6	10	10	10	8	7	10	6	10	10	10	10	10	8	2
2						1				2	1		3							
3						3					2									
[4-6]		2																		
[7-9]													1							
10																				
>10																				

Figure 87 - Number of attempts for the pick operation of the 600x400 boxes.

Place Operation involving the 400x300 boxes

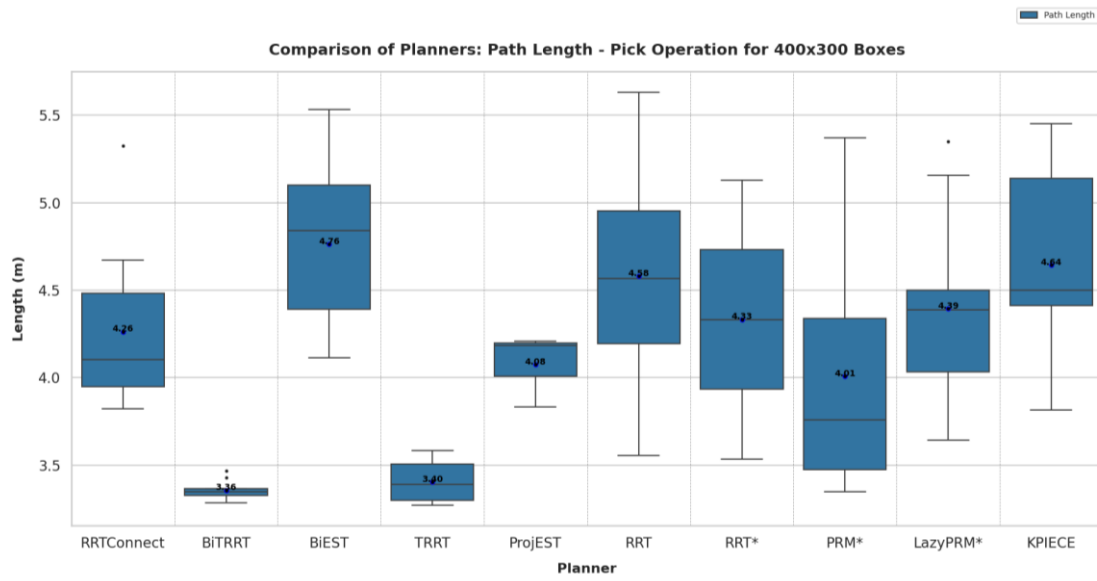
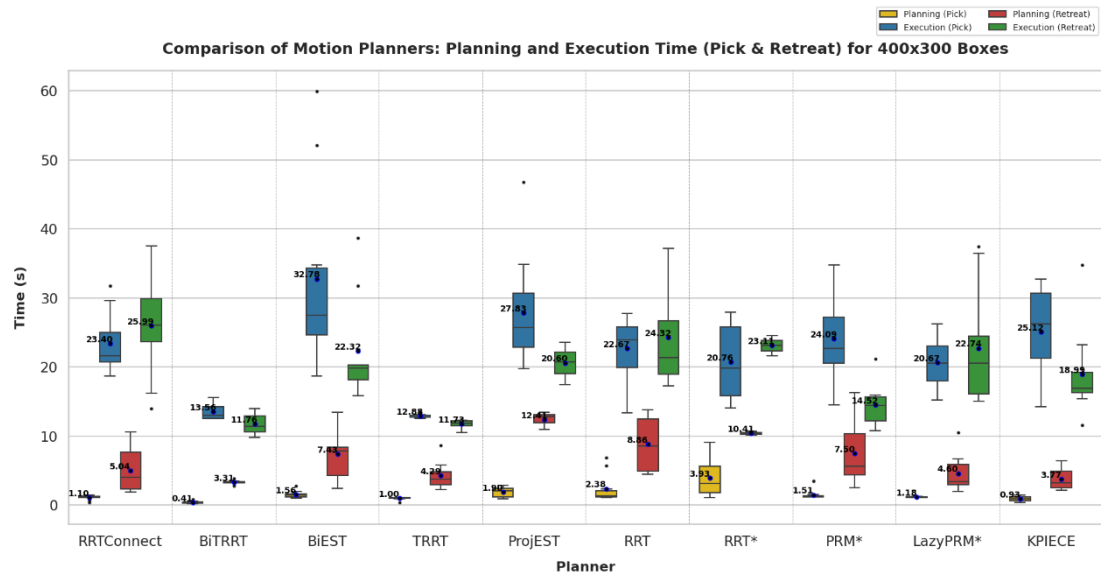


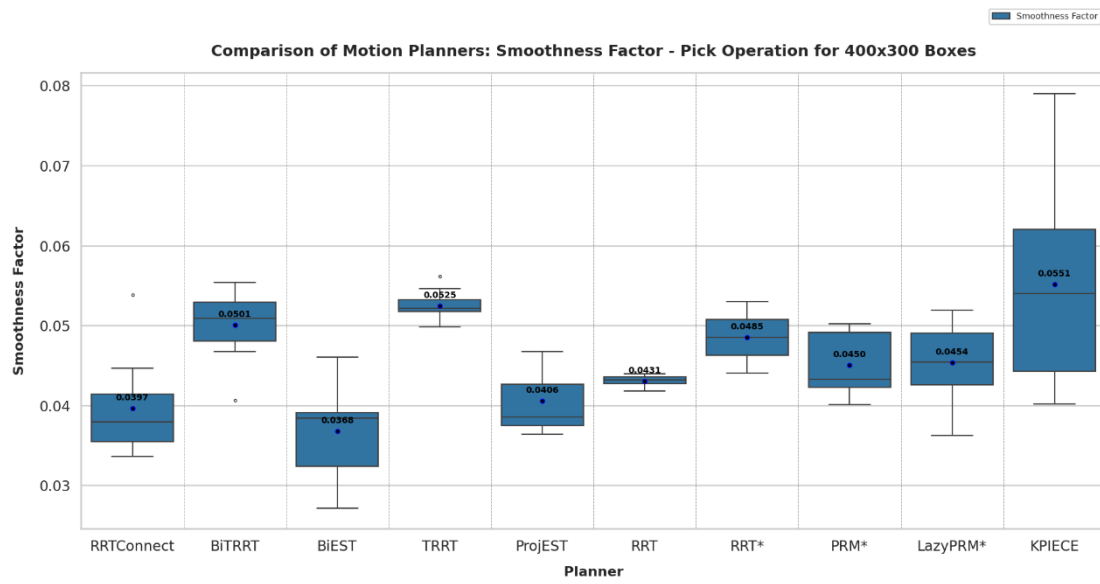


	RRT-Connect		Bi-TRRT		BiEST		T-TRT		ProjEST		RRT		RRT*		PRM*		LazyPRM*		KPIECE	
N° Attempts	place	retreat	place	retreat	place	retreat	place	retreat	place	retreat	place	retreat	place	retreat	place	retreat	place	retreat	place	retreat
1	1	5	5	9	3	8	7	8	5	10	6	10	7	10	4	10	2	9	8	10
2	1		2				1	2	3		1		2		4		3		1	
3			1				1		1		1				2		3	1		
[4-6]	2	1	1		3		1		1		2		1				5		1	
[7-9]	1				2															
10	1																			
>10	4	4	1	1	2	2														

Figure 88 - Number of attempts for the place operation of the 400x300 boxes.

Pick Operation involving the 400x300 boxes



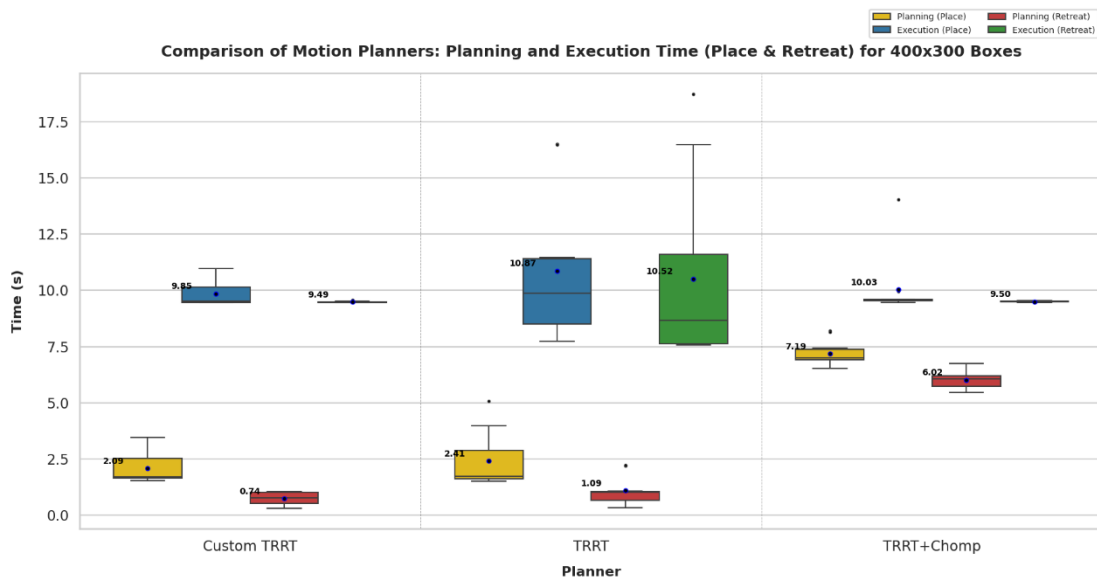
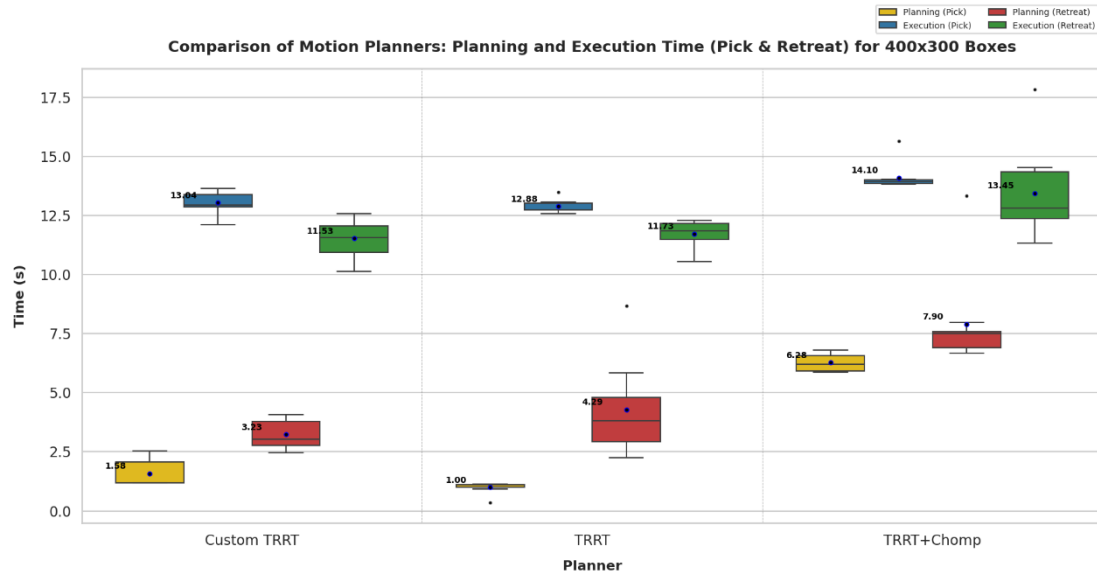


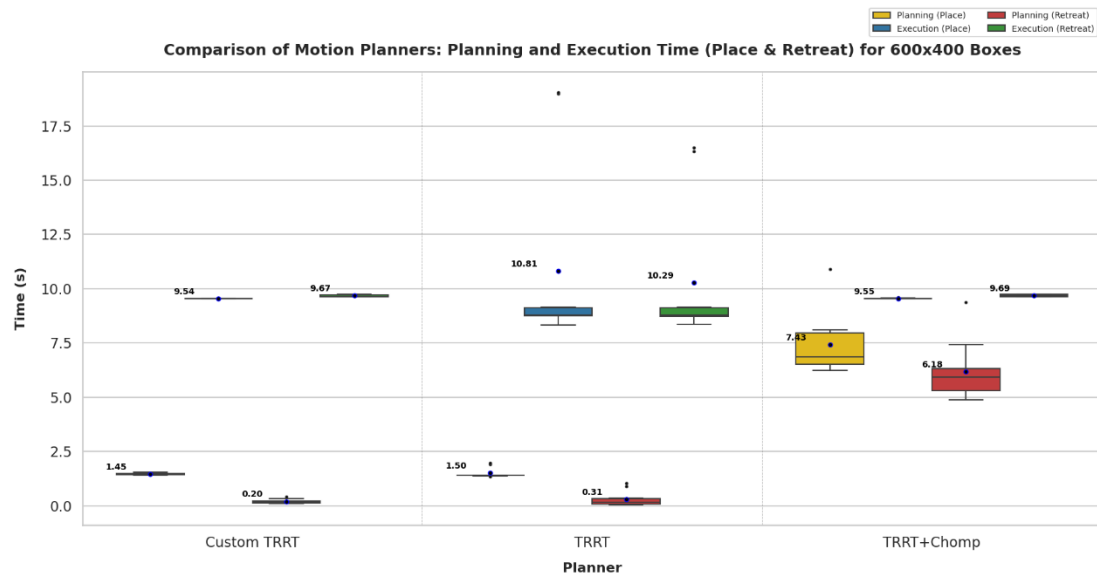
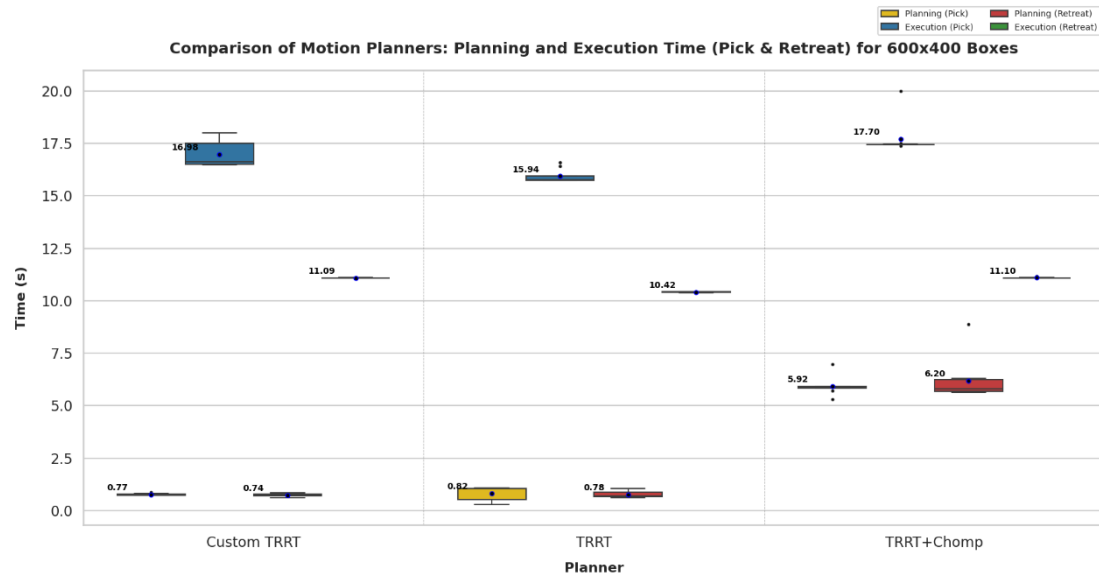
N° Attempts	RRT-Connect		Bi-TRRT		BIEST		T-TRT		ProjEST		RRT		RRT*		PRM*		LazyPRM*		KPIECE	
	pick	retreat	pick	retreat	pick	retreat	pick	retreat	pick	retreat	pick	retreat	pick	retreat	pick	retreat	pick	retreat	pick	retreat
1	10	4	10	10	10	1	10	3	5		7	1	5		9	2	10	3	10	5
2		3				2		3	5		1		2		1	1		3		1
3								2				1	1		2			1		2
[4-6]		2				4		2			2			2		2		2		2
[7-9]		1				1				1		2	2			2		1		
10						1				2										
>10						1				7		6		8		1				

Figure 89 - Number of attempts for the pick operation of the 400x300 boxes.

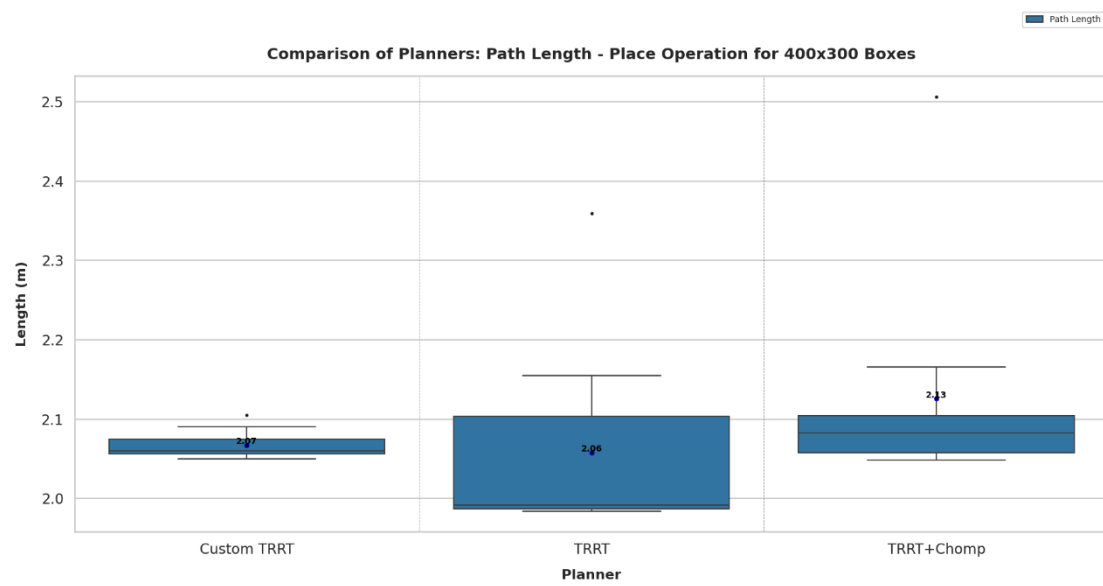
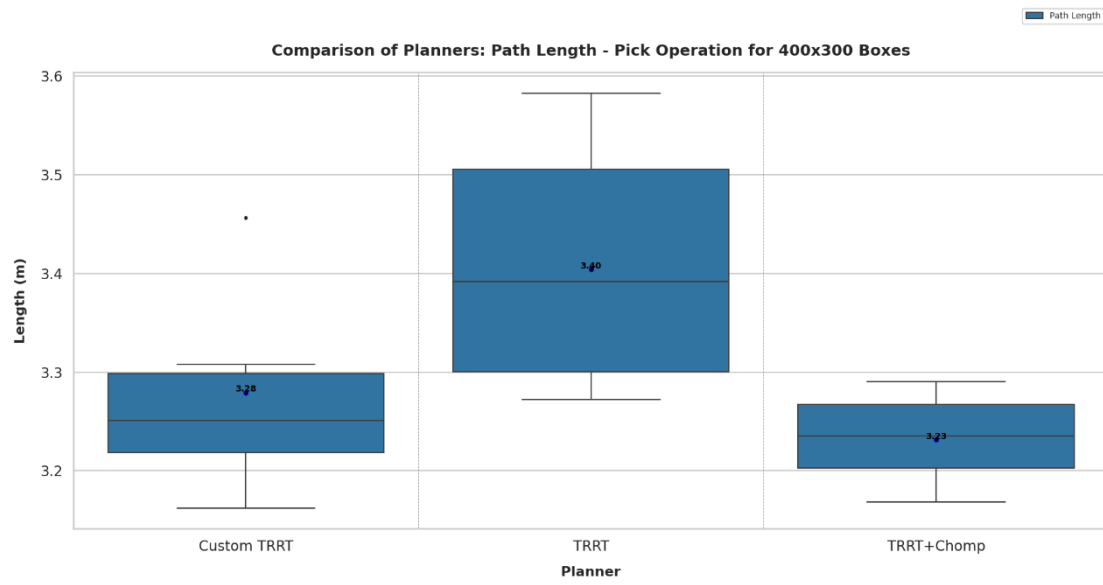
G. Performance Comparison Between Custom T-RRT Planners

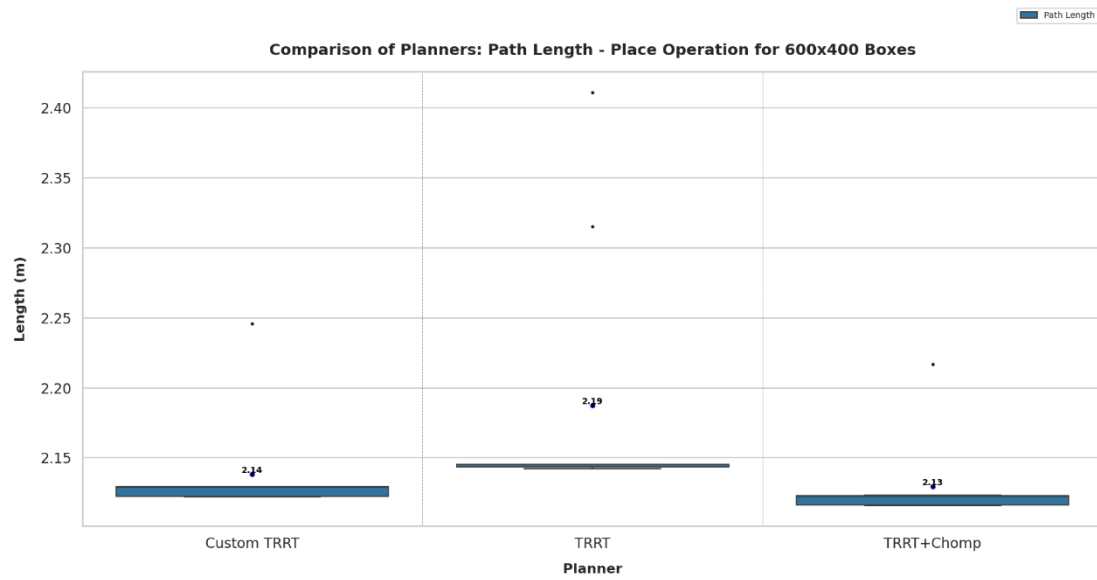
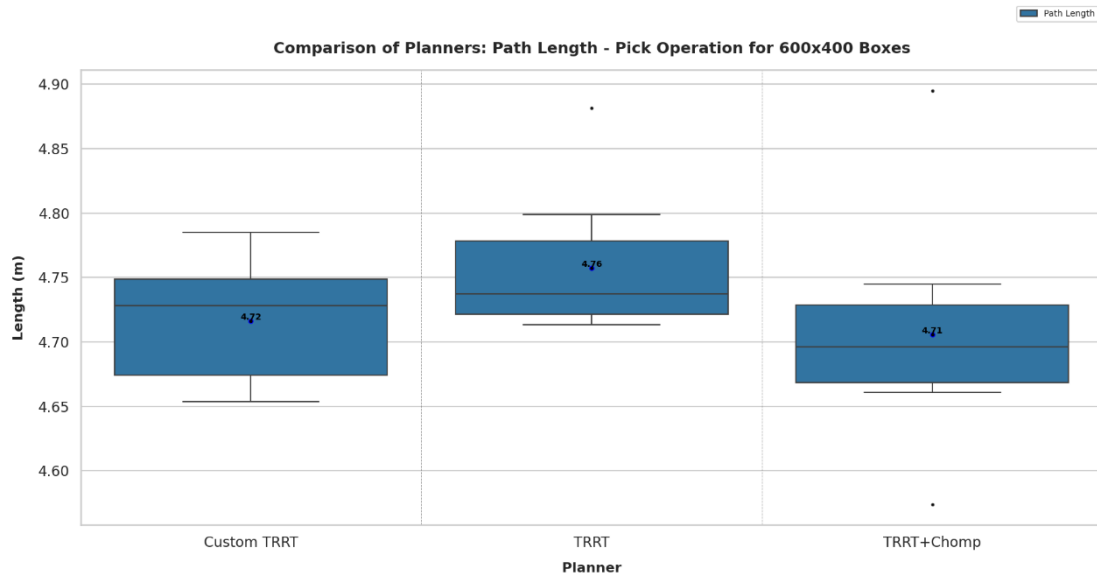
Comparison of Planning and Execution Times





Comparison of Path Length





Comparison of the Path Smoothness Metric

